Attachment 2

STATEMENT OF WORK

ACCELERATED STRATEGIC COMPUTING INITIATIVE

(ASCI)

C6939RFP6-3X

LOS ALAMOS NATIONAL LABORATORY

LOS ALAMOS, NEW MEXICO

February 12, 1996

Table of Contents

1.0 Introduction 1
1.1 Background 1
1.2 Description of Work 3
2.0 ASCI Applications Overview 5
3.0 Vision for Tera-Scale Computing 7
3.1 Hardware Environment 7
3.2 Software Environment 9
4.0 SST High-Level Technical Requirements 11
4.1 SST Hardware High-Level Requirements 12
4.2 SST Software High-Level Requirements 19
5.0 ID High-Level Technical Requirements 33
5.1 Initial Delivery Hardware Requirements 34
5.2 Initial Delivery Software Requirements 37
5.3 Performance of the ID System 42
6.0 Implementing a Sustained Stewardship TeraFLOP 46
6.1 Detailed Project Plan 46
6.2 Project Milestones 50
6.3 Performance Reviews (MR) 52
6.4 SST TeraFLOP/s sPPM Demonstration (MR) 52
Appendix A Glossary i
Hardware i
Software iii

<u>Definitions</u>
Particular paragraphs of the Statement of Work have the following designations and definitions.

(a) **Mandatory requirements designated as (MR)**
Mandatory requirements, indicated with the verb "shall", are items that are essential to the University requirements and reflect the minimum qualifications an offeror must meet in order to have their proposal evaluated further for selection (see also Attachment 4, Evaluation Criteria).

(b) **Mandatory Option requirements designated as (MO)**
Mandatory Option requirements deal with features, components, performance characteristics, or upgrades whose availability as an option is deemed a Mandatory Requirement by the University. Hence, a proposal not meeting a Mandatory Option will be deemed technically nonresponsive. Because the University may variously elect to include or exclude such options in resulting orders, each should appear as a separately identifiable item in the Price and Administration Proposal.

(b) **Target Requirements designated as (TR).**
Each paragraph so labeled deals with features, components, performance characteristics or other properties that is considered a desirable part of the ASCI system but will not be a determining factor of response compliance. Requirements in the Statement of Work indicated with the verb "may" are targets. Target Requirement responses will be considered as part of the evaluation of Technical Excellence (see Attachment 4, Evaluation Criteria).

1.0 Introduction

1.1 Background

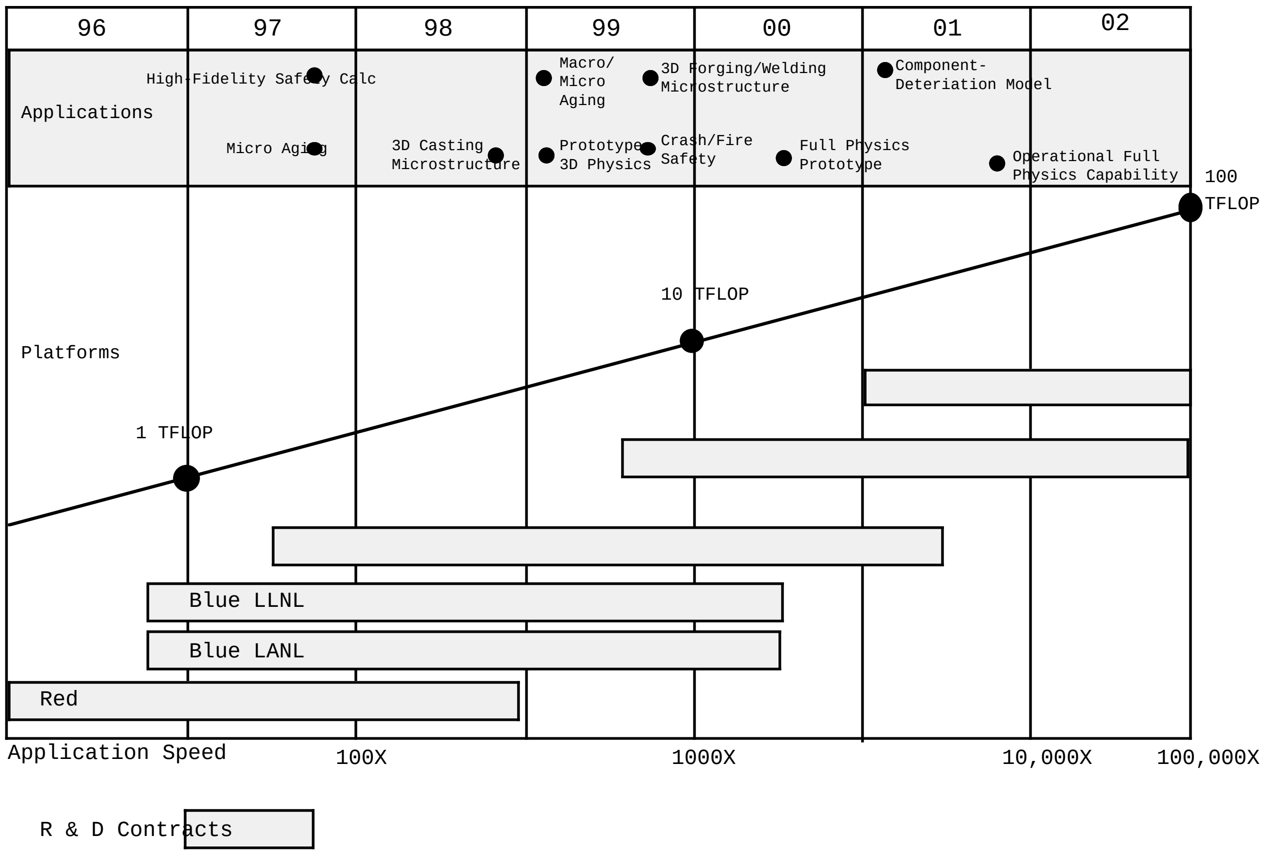
The Lawrence Livermore National Laboratory (LLNL) and the Los Alamos National Laboratory (LANL), operated by the University of California for the United States Department of Energy (DOE), as part of DOE's Accelerated Strategic Computing Program (ASCI, which also includes Sandia National Laboratories) are jointly seeking to accelerate the development of Tera-scale computing capability on clusters of shared memory multiprocessors (SMPs). This activity is known as ASCI Blue.

This effort is part of the focused DOE Accelerated Strategic Computing Initiative of the Stockpile Stewardship Program. Because of the US commitment to ending underground nuclear testing, dramatic advances in computer technology are required to make virtual testing and prototyping viable alternatives to traditional nuclear and non nuclear test-based methods.

To economically accommodate full support of new 3-D massively parallel applications we require the achievement of sustained TeraFLOP/s performance for a representative Stockpile Stewardship problem on a scalable cluster of SMPs, a scalable software/middleware environment that supports large-scale code execution and development, a high bandwidth, low latency cluster interconnect, and siting of equipment at one of the Laboratories for at least two years.

The US commitment to ending underground nuclear testing, constraints on non-nuclear testing, and loss of production capability call for new means of verifying the safety, reliability, and performance of the US nuclear stockpile. One of these means is compute-based virtual testing and prototyping of nuclear weapon systems. ASCI is one element of DOE’s Stockpile Stewardship Program, designed to advance DOE/Defense Programs computational capabilities to help meet the future needs of stockpile stewardship. ASCI will create the leading-edge computational modeling capabilities that are essential for maintaining the safety, reliability, and performance of the US nuclear stockpile in the absence of underground testing.

ASCI applications require immediate computational throughput gains of 100 or more over current supercomputer capability and future enhancements well beyond the TeraFLOP/s range over the next several years.



It is important to emphasize that the driving force for ASCI is the Stockpile Stewardship Program. This motivates the need for the development of suitable application codes, the supporting computational problem solving environment, as well as hardware platforms upon which the applications can be executed.

The purpose of this activity is the realization of large scale numerical simulations sufficient to support the Stockpile Stewardship Program. Although the ultimate goal of the ASCI program is a computational capability far beyond TeraFLOP/s, the initial research objective targets a performance level of one to tens of TeraFLOP/s sustained performance on ASCI applications. The culmination of the R &D activity will be a reliable, robust system capable of delivering Tera-scale performance on large scale ASCI applications.

However, this is not a standard hardware procurement, but rather Subcontractor research and development. The Subcontractor's ability to provide the very high performance computing capability necessary to support Science-Based Stockpile Stewardship (SBSS) computing applications, together with the University's expertise in applications development, is expected to produce the desired result: Tera-scale computing. Accomplishing the aggressive ASCI goals will require significant monetary investment by the University and execution of a highly complex research and development strategy by the Subcontractor.

1.2 Description of Work

The objective of this R & D activity is the development of a scalable SMP cluster-based system, including all hardware and software, that will provide sufficient capacity and capability to develop, debug, and execute large simulation codes at the TeraFLOP/s performance level in support of SBSS.

The Subcontractor shall:

1.2.1 Detailed Project Plan (MR)
Provide a detailed project plan for the ASCI Blue development project.

1.2.2 Execute Development Plan (MR)
Accomplish the milestones of the Subcontract; accelerate the development of hardware and software for the purpose of achieving the stated Tera-scale performance goals.

1.2.3 Install ID System(s) (MR)
Configure, install, support and maintain the Initial Delivery (ID) system, including hardware and software, at LANL or LLNL. At the option of the University, install an additional ID system.

1.2.4 Technology Refresh (TR)
Provide hardware and software technology refresh to those systems as available.

1.2.5 On-site Support (MR)
Provide on-site assistance to Laboratory applications developers.

1.2.6 Scalable Development Environment Goal (MR)
Demonstrate the scalability of essential software capabilities in the application development environment across the clustered SMP system.

1.2.7 Sustained TeraFLOP Performance Goal (MR)
Demonstrate at least a sustained one (1.0) TeraFLOP/s performance on the sPPM Science-Based Stockpile Stewardship application.

1.2.8 Three Peak TeraFLOP Performance Goal (MR)
Demonstrate a machine with the sum of peak and sustained performance of at least four (4.0).

1.2.9 Install Sustained Stewardship TeraFLOP (SST) System (MR)
Configure, install, support, and maintain the TeraFLOP/s scalable clusters at LANL or LLNL.

1.2.10 Memory Upgrade (MO)

At the option of the University, upgrade the TeraFLOP/s scalable clusters as described in paragraph 6.2.8.

1.2.11 Performance Reviews (MR)

Apply Laboratory-provided metrics (including quarterly reviews) during the course of the contract to gauge progress.

1.2.12 Successful Project Completion (MR)

Demonstrate the achievement of the development objectives at the conclusion of the contract.

End of Section 1

2.0 ASCI Applications Overview

ASCI applications codes will generally perform complex time-dependent three-dimensional simulations of multiple physical processes, where often the processes are tightly coupled and will require physics models linking microscale phenomena to macroscopic response. These codes will address nuclear weapons issues relating to safety, performance, aging, and remanufacturing which are of interest to the DOE Weapons Laboratories in support of SBSS. The ASCI codes will include multi-material shock hydrodynamics, radiation and particle transport, atomic physics, material properties, and structural response.

Demanding stockpile stewardship issues are expected to drive intense development of ASCI applications codes and advanced numerical algorithms and will require innovative software techniques to achieve the necessary delivered performance on advanced computing platforms. Current application characteristics and expected trends are described below, although ASCI applications codes and numerical algorithms are expected to evolve rapidly.

Current ASCI applications are generally based on finite difference algorithms on 3D geometries represented by regular Cartesian, logically-regular hexahedral, or unstructured tetrahedral meshes. Regular mesh codes are likely to include time-dependent adaptive mesh refinement (AMR) whereas unstructured mesh codes already support reconnections. Finite element-based applications are possible as well. Time-dependent simulations will be the norm. They include explicit, and coupled explicit-implicit algorithms. Many applications will simulate multiple coupled physical processes collectively in time. Some of the processes may be localized to particular spatial domains or even in time. There will be applications that simulate continuous fields and some that simulate discrete particles or states.

Multi-material hydrodynamics codes based on Eulerian, Lagrangian, and arbitrary-Lagrangian-Eulerian (ALE) formulations tend to be explicit in time, use nearest-neighbor spatial differencing, and employ sophisticated material property models. Eulerian codes usually have interface tracking and reconstruction. Multiple strategies for AMR support in Eulerian codes are being explored. The radiation and particle transport methods under investigation include diffusion, P_n, simplified P_n, discrete ordinates, and Monte Carlo formulations. The diffusion, P_n, and simplified P_n algorithms all resemble typical standard finite difference diffusion methods. Discrete ordinates algorithms may involve a direct-solve wavefront solution strategy. The Monte Carlo algorithms involve discrete particle tracking. Atomic physics codes involve electronic state calculations. Material properties codes involve molecular dynamics style simulations at the microscopic or grain level. Structural codes handle material deformation and flow, energy dissipation, and stress and strain modeling at the macroscopic scale and are typically finite element based.

The major emphasis for ASCI applications will be on the design, development, refinement, and execution of parallel 3D codes on large parallel computer systems; in addition, they may be portable to machines as small as single processor workstations to facilitate code development. The codes will be written in standard Fortran 90, Fortran 77, C, and C++, with this ordering reflecting the expected usage of each language. Mixtures of these languages, especially Fortran and C, are typical. Use of typical extensions to Fortran 77, such as automatic arrays, pointers, and MIL STD 1753 extensions, are common. Parallel code implementations are expected to use both explicit domain decomposition with MPI message passing and some form of shared memory multitasking on SMP systems. It is also likely that some codes will use MPI message passing simultaneously both within an SMP and across SMPs. Access to shared memory multitasking is expected to be through automatic or directive-based compiler support or through system calls for explicit thread management and control (e.g., POSIX threads). Some exploratory usage of High Performance Fortran (HPF) or of parallel C languages might also occur.

End of Section 2

3.0 Vision for Tera-Scale Computing

3.1 Hardware Environment

The approach taken by this effort is based on providing the requisite computational resources through clusters of shared-memory multi-processor systems, or so-called SMPs. We view clustering of SMPs as an economically viable mechanism for achieving the multiple TeraFLOP/s levels of performance required by stockpile stewardship applications. We do not necessarily mean to indicate by SMP the narrower concept of bus-based Symmetric Multi-Processors. Rather, our concept of clusters of SMPs has six components to it:

- a **hierarchical** system of memories and latencies;
- multiple high-performance **distributed** systems;
- **shared-memory** over a significant and economical computational resource with low-latency, high-performance memory access;
- **multi-processor**, supporting a shared-memory environment across multiple processors;
- **commodity priced** SMPs which represent high-end systems of a regular commercial product line (they are not special purpose designs);
- **commodity priced** (third-party) **peripherals**.

Shared-memory, high-end, microprocessor-based, multi-processor compute servers in the context of this activity are typical examples of the envisioned SMP machines.

The emerging clustered SMP computing environment will ultimately support a hierarchical distributed computing paradigm covering the full range of the memory and latency hierarchies encountered in clusters. We envision these SMP clusters to be assembled from building blocks whose size is determined by market economics. The approach we wish to take extends beyond the MPP strategy for large scale parallelism in which up to thousands of processors are interconnected in a homogeneous, flat, distributed memory system in which inter-CPU communication is handled by message passing. This ASCI activity will accommodate applications targeted for MPPs but its real advantage lies in the ability to exploit the computational power of multiple system aggregation strategies.

The ideal size for the SMP building block in this approach is determined by a balance of cost effectiveness, reliability, hardware/software scalability, the number of processors per SMP and number of SMPs in the clusters. We anticipate that this balance may change over the lifetime of the contract. Our preference is for the capability on ASCI applications of a single SMP to be as large as economically viable.

Clusters of SMP systems that can address the anticipated stockpile stewardship application requirements are anticipated to scale according to the following ratios. These ratios were developed through the Laboratories' multi-year experience with addressing challenging computational problems on high performance computing systems from a variety of application domains.

1 FLOP/s peak performance /
.5 - 1 byte memory size /
10 -100 byte disk storage /
4-16 byte per second L1-L2-cache bandwidth /
1-3 byte per second memory bandwidth /
0.1-1.0 bit per second communications bandwidth /
0.01 - 0.1 byte per second disk bandwidth

Building and delivering a Tera-scale computing resource is a daunting task. Within the context of a research and development contract it is anticipated that a well balanced hardware approach will follow the following four notional phases: 1) an Initial Delivery (ID) system for ASCI application code development; 2) technology refresh as the ID system ages and fruits of the ASCI project become available; 3) one sPPM-sustained TFLOP/s system with a peak performance at a higher level; and 4) memory upgrade.

Due to SBSS programmatic requirements, the University has developed a general schedule for delivery of hardware and software assets. Of prime consideration are the decoupling of compute and memory SST deliveries and the timeliness of an Initial Delivery (ID) system for code development.

Target Delivery Date	System	Metrics
90 days after contract award	Initial Delivery (1st System)	100 GFLOP/s Peak Performance, 50 GB Main Memory, 2.5 TB RAID Disk
90 days after contract award	Initial Delivery (Optional 2nd System)	100 GFLOP/s Peak Performance, 50 GB Main Memory, 2.5 TB RAID Disk
2Q CY 1998	SST Delivery	1.0 sustained sPPM TFLOP/s performance. Memory to Peak FLOP/s ratio is 0.167. 75 TB RAID Disk
2Q CY 1998	SW Scalability	Scalability of the application development environment tools across the clustered SMP system.
Q1 CY 1999	Memory Upgrade	Increase memory to Peak FLOP/s of SST from 0.167 up to 0.5.

3.2 Software Environment

The spirit of ASCI is “One Program—Three Labs.” All three National Laboratories will participate in applications development, testing and applying the resources. This has tremendous impact on the software environment which, of course, is the key to the success of this activity. Effectiveness is expected to occur through system integration and the development of “middleware” software which allows clusters of SMPs to effectively support a single computation. Hardware transparency is essential in many areas if we are to preserve the investment in the applications beyond the lifetime of the hardware.

As has been previously stated, the objective of this requirement is to provide a platform for ASCI applications that is over two orders of magnitude more powerful than that available at LANL and LLNL today. To put it more directly, the purpose of this activity is to provide a Tera-scale computing environment for the very demanding applications required for Science-Based Stockpile Stewardship. A truly usable Tera-scale computing resource requires a software environment that scales as well as the hardware and provides a rich scalable code development environment. This point can not be overemphasized.

The University is aware that a Tera-scale software environment is an enormous expectation. The software requirements are aimed at setting out our vision of what is necessary to achieve the ASCI program goals. However, just as in the case of the hardware, the very high performance computing market must significantly stretch to meet these objectives. Therefore, it is our strategy to encourage the Subcontractor to accelerate software development necessary to achieve these objectives. It is anticipated that activities related to this Subcontract will have at least three general phases. These phases are purely notional (i.e., they don’t correspond to contract milestones) and are intended for informational purposes only.

In the first phase, there will be intensive applications code development on the ID system. In addition, it is anticipated that there will be a Subcontractor effort to advance the code development environment and system management tools. In this phase basic ASCI applications development will take place either within an SMP utilizing the SMP programming paradigm (explicit parallelism exploited with POSIX threads or implicit parallelism exploited by the compiler) or utilizing message passing via MPI both intra-SMP and intra-cluster. The choice will depend on the best implementation for specific physics packages. ASCI applications will be combinations of many physics packages and will contain both styles of parallelism. On the systems side, it is anticipated that development will be needed to assure robustness and extend the features of the code development environment on a single SMP. This will include compilers, loaders, debuggers, performance analysis tools, gang scheduling, resource management, DCE and system administration to name but a few. This is mostly an interactive code development environment.

In the second phase, it is anticipated that the applications will need more “batch” computing time as well as interactive code development cycles. Development, on the system side, will be extending the cluster notion to provide more of a “single system image” and operation as a single unit. Basic development of SMP cluster wide reliability, availability and serviceability (RAS), gang scheduling, resource management and performance analysis tools will be extended from an SMP-specific focus to cluster-wide. To give a specific non-trivial example, it is anticipated that a cluster wide process and session identification space will have to be developed to support the single system image, gang scheduling and resource management.

In the third phase, it is anticipated that most of the exploratory code development environment and the system software development will be completed. It is in this phase that the software productization activities will dominate. That is to say, the third phase will be dominated by extending and productizing the good ideas (and implementations) and discarding the weak ones.

Having given this general anticipated progression of activities, the Subcontractor shall be responsible for clearly delineating a coherent and credible software development strategy to achieve the ASCI project goals.

End of Section 3

4.0 SST High-Level Technical Requirements

The end product of the ASCI Blue research and development activity is a well balanced compute resource over one hundred times more powerful than those currently available at either LLNL or LANL. It will be focused on solving the initial critical problems, that is, the large-scale application problems at the edge of our understanding of weapon physics. This Sustained Stewardship TeraFLOP (SST) system must be useful in the sense of being able to deliver a large fraction of peak performance to a diverse scientific and engineering workload. It must also be useful in the sense that the code development and production environments are robust and facilitate the dynamic workload requirements.

The specifications below define a Sustained Stewardship TeraFLOP/s SMP scalable cluster based on the performance of the sPPM demonstration code identified in Section 6.4. Obviously, the Subcontractor will necessarily have to estimate the efficiency of sPPM on the proposed system in order to determine what to actually bid, price and ultimately deliver to meet the mandatory requirement identified in Section 4.1.1.1. If the efficiency of sPPM on the proposed system is below 33% then more than three peak TeraFLOP/s of computational resources will be required. However, if the sPPM efficiency is greater than 33%, the Subcontractor will deliver less than three peak TeraFLOP/s to meet section 4.1.1.1. In any event, the sum of peak plus sPPM sustained performance must be at least 4.0 TeraFLOP/s. In the University's view, this issue will motivate additional Subcontractor innovation during contract execution.

Due to the classified and unclassified ASCI programmatic requirements the SST system must function in the classified (RED) and unclassified (BLACK) network environments. In addition, we anticipate the frequent (on the order of once a week) switching between the RED and BLACK environments. We require that the transition time between RED and BLACK (and vice-versa) be less than sixty minutes. The intense code development effort involved in the ASCI program requires a <u>stable</u> ASCI code development environment (CoDE) be available concurrently in the RED and BLACK network environments. To be useful, this CoDE must have a minimum representation of all the key hardware features of the larger migratable SST compute resources and allow for the full application development cycle (edit, compile, link and debug). These requirements imply that no CPU or SMP local disks may be employed. The disk and external networking requirements in this Statement of Work are for the aggregate of both RED and BLACK network environments.

Development of the SST shall comply with the requirements identified in section 6.0, Implementing a Sustained Stewardship TeraFLOP/s.

The specific hardware and software Mandatory Requirements the SST system shall meet are delineated in sections 4.1 and 4.2 with (MR) designation.

In addition to the mandatory hardware and software requirements, the Subcontractor may deliver any Target Requirements (TR) for the SST, and any additional features consistent with the objectives of this project and Subcontractor's Research and Development Plan, which the Subcontractor believes will be of benefit to the project.

4.1 SST Hardware High-Level Requirements

4.1.1 Scalable Cluster of SMPs

4.1.1.1 Sustained Stewardship TeraFLOP SMP Scalable Cluster (MR)

The Subcontractor shall provide a Sustained Stewardship TeraFLOP/s (SST) system composed of multiple Shared memory Multi-Processors (SMPs) connected via a scalable intra-cluster communications technology. The system shall have a peak plus sustained performance of at least four (4.0) TeraFLOP/s and sustain at least one (1.0) TeraFLOP/s (1.0×10^{12} floating point operations per second) on the sPPM benchmark.

Example: If “p” is the peak performance of the system and “s” is the sustained on sPPM performance of the system, then the above mandatory requirement states that

$$s \geq 1, \text{ and } p+s \geq 4.0.$$

If we define the machine efficiency as $e = s/p$, then the above equations become:

$$p(e) \geq 1/e \qquad \text{for } e < 1/3$$
$$\text{and}$$
$$p(e) \geq 4.0/(1.0+e) \qquad \text{for } e \geq 1/3$$

Hence,

$$p(1/2) \geq 4/1.5 \approx 2.67,\ p(1/3) \geq 3.0 \text{ and } p(1/5) \geq 1/0.2 = 5.0$$

4.1.1.2 SST Component Scaling (MR)

In order to provide the maximum flexibility to the Subcontractor in meeting the goals of the ASCI project the exact configuration of the SST SMP scalable cluster is not specified. Rather the SST configuration is given in terms of lower bounds on component attributes. The SST SMP scalable cluster configuration shall meet or exceed the following parameters:

- Memory Size ≥ 0.5 TB
- Disk Space ≥ 75 TB
- Cache Bandwidth ≥ 12 TB/s
- Memory Bandwidth ≥ 3 TB/s
- Intra-Cluster Network Bi-Section Bandwidth ≥ 1.5 Tb/s
- System Peak Disk I/O Bandwidth ≥ 90 GB/s

4.1.1.3 SST Applications Memory (MO)

The Subcontractor shall install at least 1.5 TB of memory in the SST as an option.

4.1.1.4 Cluster Wide High Resolution Event Sequencing (TR)

The SST may include hardware support for a cluster-wide real-time clock or other hardware mechanism for cluster-wide event sequencing. The resolution of this mechanism may be less than 1.0 micro-second ($1x10^{-6}$ seconds). This facility would be used for parallel program debugging and performance monitoring.

4.1.1.5 Cluster Interconnect Reliability and Performance (TR)

If interconnect components fail (e.g., SMP network interface, network stage link, router or switch), each SMP may still be able to communicate with all other SMPs over the cluster interconnect. That is, there may be multiple interconnect paths or routes between SMPs so that, in the event of interconnect component failure, each SMP can still communicate with all other SMPs. The SMP interconnect may have the ability to segregate network traffic so that operating system related traffic (e.g., I/O service, OS to OS communication) does not interfere with user data communication traffic.

4.1.1.6 Cluster Interconnect Link Bandwidth (TR)

The network bandwidth available to each SMP may be capable of sustaining at least 0.1 GB/s per peak GFLOP/s of processing power.

Example: If the Subcontractor proposes 59 SMPs each with 64 processors rated at 0.8 GFLOP/s peak, then each SMP link bandwidth may be rated at 64*0.8*0.1 GB/s = 5.12 GB/s. This would require 51.2 HIPPI (800 Mb/s) ports.

4.1.1.7 Cluster Interconnect Latency (TR)

The cluster interconnect latency, as measured by sending a minimum length MPI message from user program memory on one processor in the SMP cluster to user program memory on any other processor in the cluster and receiving back an acknowledgment divided by two (standard MPI user space ping-pong test), may be less than 5.0 micro-seconds ($5x10^{-6}$ seconds).

4.1.1.8 Remote Memory Access (TR)

The system may include support for remote memory reads and writes. A remote memory read returns the same data that a processor residing in the SMP with the remote memory location would have seen if it had read the location at the same time. Similarly, a remote memory write updates the location in a remote memory in the same way that a local processor memory write would have done. This remote memory access mechanism may include hardware memory protection that protects the memory space of each individual user from all other users within the cluster.

4.1.2 Shared Memory Multi-Processor

4.1.2.1 SMP Platform (MR)

The Shared memory Multi-Processor (SMP) platforms shall be a set of CPUs sharing random access memory within the same memory address space. The CPUs shall be connected via a high speed, extremely low latency mechanism to the set of hierarchical memory components. The memory hierarchy consists of at least processor registers, cache and memory. The cache may also be hierarchical. If there are multiple caches, they shall be kept coherent automatically by the hardware. The main memory may be a Non-Uniform Memory Access (NUMA) architecture. The access mechanism to every memory element shall be the same from every processor. More specifically, all memory operations shall be accomplished with load/store instructions issued by the CPU to move data to/from registers from/to the memory.

4.1.2.2 CPU Characteristics (MR)
Each SMP shall be an aggregate of homogeneous general purpose computers (CPUs) consisting of high-speed arithmetic, logic units, and memory, together with the necessary control circuitry and interprocessor communications mechanism(s). Each shall execute fixed and IEEE 754 floating-point arithmetic, logical, branching, index, and memory reference instructions. A 64-bit data word size shall directly handle IEEE 754 floating-point numbers whose range is at least 10^{-305} to 10^{+305} and whose precision is at least 14 decimal digits. The CPUs and memory hierarchy shall provide an appropriate mechanism for interprocessor communication, interrupt, and synchronization.

4.1.2.3 Minimum CPU Performance (TR)
The minimum single CPU performance, as measured by peak performance, may be at least five hundred million 64-bit floating point operations per second (500 MFLOP/s).

4.1.2.4 Minimum SMP Performance (TR)
The minimum SMP performance, as measured by peak performance, may be at least six billion 64-bit floating point operations per second (6 GFLOP/s).

4.1.2.5 IEEE 754 32-Bit Floating Point Numbers (TR)
The CPUs may have the ability to operate on 32-bit IEEE 754 floating-point numbers whose range is at least 10^{-35} to 10^{+35} and whose precision is at least 6 decimal digits, for improved memory utilization and improved execution times.

4.1.2.6 Test-And-Set Instruction (TR)
The Subcontractor may provide sufficient atomic instructions (e.g., test-and-set or load-and-clear) along with some atomic incrementing instruction (e.g., test-and-add or fetch-and-increment/fetch-and-decrement) so that the usual higher level synchronizations (i.e., critical section, barrier, etc.) can be constructed. Additionally, these synchronization instructions or their higher-level equivalents may be directly accessible from user programs.

4.1.2.7 Programmable Clock (TR)
There may be a real-time clock per CPU capable of causing a hardware interrupt after a preset interval (i.e., a programmable clock). The clock frequency may be at least one megahertz and the preset interval may be capable of being set in increments of 10 microseconds or less. There may be at least 16 seconds allocated for the time interval. This clock may have at least 24 bits.

4.1.2.8 Hardware Interrupt (TR)
The SMP may have hardware support for interrupting given subsets of computational processors based on conditions noted by the operating system or by other computational processors within the subset executing the same user application.

4.1.2.9 Hardware Performance Monitors (TR)
The CPUs may have hardware support for monitoring system performance. As much of this data as possible may be made available directly to applications programmers and to code development tools. Memory hierarchy behavior, counting floating point operations and message passing performance are of particular interest.

4.1.2.10 Hardware Debugging Support (TR)
The CPUs may have hardware support for debugging, and in particular, hardware that enables setting data watch points (e.g., hardware interrupts on read/write to a specific virtual memory location).

4.1.3 Memory Hierarchy

4.1.3.1 Shared Main Memory (MR)
The main memory (as distinct from cache memory) shall be high-speed, single-byte addressable, random access, single-bit correcting, double-bit detecting (SECDED). All components of the main memory (e.g., local and remote) shall be addressable by a single mechanism, (i.e., load/store instructions), from all compute CPUs in the SMP. Memory which is directly addressable by only a subgroup of the processors shall be part of a cache, if present. Note that this does not require that the distance (as measured in latency or bandwidth) to every memory element be the same from every processor.

4.1.3.2 Memory Hierarchy Latency (TR)
Each SMP may include a memory hierarchy of at least processor registers; cache; main memory. The cache may be hierarchical. The main memory may be a Non-Uniform Memory Access (NUMA) architecture. The load and store latency between processor registers and the slowest cache component may be no larger than 10 processor clocks. The load and store latency between processor registers and main memory may be no larger than 100 processor clocks if the slowest cache component is at least 4.0 MB in size and no larger than 20 clocks if the cache is less than 4.0 MB in size. If main memory is NUMA, then this requirement is between the processor registers and the most distant (slowest) memory element in the SMP address space, assuming a best case memory access scenario.

4.1.3.3 Additional Physical Address Space (TR)
Each SMP may have the capability to directly address at least ten (10.0) TB of physical memory with at least 44 bits of physical address space.

4.1.3.4 Memory Consistency Model (TR)
Within an SMP, a weak memory consistency model is allowed if there are specific hardware constructs (available to application programs) that can enforce a strong

(sequential) memory consistency model for sequences of instructions (i.e., synchronization). This is sometimes known as a weak memory reference ordering model with additional hardware support for synchronization.

4.1.4 RED/BLACK Resource Usage Model

4.1.4.1 RED/BLACK Code Development Environments (MO)

The Subcontractor shall provide sufficient resources to support independent concurrent classified and unclassified code development environments (CoDEs). The majority of the SST compute resource shall migrate between the two CoDEs. Each CoDE shall consist of some portion of the disk I/O and external networking resources. The CoDE shall have sufficient computational capability to directly compile and debug applications that implement the full hierarchical distributed memory model (i.e., it must have representative portions from the entire memory latency and bandwidth hierarchy).

4.1.4.2 RED/BLACK I/O Resources (MR)

The disk space provided to meet requirement 4.1.1.2 shall be split into two separate independent, external network and disk I/O subsystems (one RED and one BLACK). The aggregation of these two I/O environments are used to compute the total external network interface and disk resources provided. The SMP cluster shall be able to boot from either of these I/O subsystems.

4.1.4.3 RED/BLACK Migration (MR)

The SST compute resources shall migrate between classified (RED) and unclassified (BLACK) usage in less than sixty (60) minutes. The migration shall consist of at least the following steps: SST compute resource shutdown (and possible power cycling when migrating from RED to BLACK); the I/O and external network interface resources associated with one classification environment be physically decoupled from the SMP cluster and the external network interface resources associated with the other classification environment reconnected; and reboot the SST compute resources. Decoupling shall be by a physical separation of at least 1.0 foot. There shall be no writeable (by any user other than privileged users) non-volatile memory in the system except in the detachable, independent RED/BLACK file storage subsystems.

4.1.4.4 SST RAID Arrays (TR)

All disk resources provided may be RAID 3 or RAID 5 arrays. The RAID units may have high availability characteristics. These include redundant failover power supplies and fans, hot swapable disks, the capability to run in degraded mode (one disk/RAID string failure), and the capability to rebuild a replaced disk on the fly with minimal performance impact.

4.1.4.5 Single Sustained I/O Bandwidth (TR)

The minimum sustained transfer rate for reading or writing of a single 1.0 GB file to or from a single logical file system may be no less than 20 MB/s.

4.1.4.6 Parallel Sustained I/O Bandwidth (TR)

The minimum sustained parallel I/O transfer rate for reading or writing of a parallel 10.0 GB file to a parallel file system may be no less than 0.002 Byte/s per peak FLOP/s.

Example: A 3.0 TFLOP/s SST system may deliver 0.002*3000 = 6.0 GB/s parallel I/O bandwidth to a single application running on the entire SST. This would allow a code to write a 0.5 TB restart dump in a little under 2 minutes.

4.1.4.7 Cluster High Speed External Network Interfaces (TR)

The SST system may include a minimum of 0.00125 Bit/s per peak FLOP/s of aggregate external networking bandwidth. The system may also include at least 4 HIPPI connections (2 for the RED partition and 2 for the BLACK). The Subcontractor may work with the University to assure these connections will be capable of interoperating with existing or future LANL or LLNL systems and network gateways.

Example: A 3.0 TFLOP/s SST system may have 0.00125*3,000,000 = 3750 Mb/s external network bandwidth. This can be accomplished with approximately 6 OC-12 (650 Mb/s) external network interfaces.

4.1.4.8 Cluster I/O Upgradeability (TR)

The disk capacity may be field upgradeable to twice the initial system capacity.

4.1.5 Reliability, Availability, Serviceability and Maintenance

4.1.5.1 Power Cycling (TR)

The system may be able to tolerate power cycling at least once per week over its life cycle.

4.1.5.2 Hot Swap Capability (TR)

Hot swapping of failed Field Replaceable Units (FRUs) may be possible without power cycling the cabinet in which the FRU is located The maximum number of components (such as CPUs, memory, disks and power supplies) contained in or on one FRU may be less than 1% of the components of that type in the system.

4.1.5.3 Production Level System Stability (TR)

The system (both hardware and software) may execute 100 hour capability jobs (jobs exercising at least 90% of the computational capability of the system) to successful completion 95% of the time.

4.1.5.4 System Down Time (TR)

Over any four week period, the system may have an effectiveness level of at least 95%. The effectiveness level is computed as the average of period effectiveness levels. A new period of effectiveness starts whenever the operational configuration changes (e.g., a component fails or a component is returned to service). Period effectiveness level is computed as University operational use time times max[0, (p-2d)/p] divided by the period wall clock time. Where p is the number of CPUs in the system and d are the number disabled. Scheduled Preventive Maintenance (PM) is not included in University operational use time.

4.1.5.5 FRU Diagnostics (TR)

Diagnostics may be provided that, at a minimum, isolate a failure to a single FRU. This diagnostic information may be accessible to operators through networked workstations.

4.1.5.6 Failure Isolation Mode (TR)

SMP (or FRU) failures may be able to be determined, isolated, and routed around without system shutdown. The operators may be able to reconfigure the system to allow for continued operation without use of the failed SMP (or FRU). The capability may be provided to perform this function from a remote network workstation.

4.1.5.7 Scalable System Diagnostics (TR)

There may be a scalable diagnostic code suite that checks processor, cache and RAM memory, network functionality, and I/O interfaces for the full system in less than 30 minutes.

4.1.5.8 System Graceful Degradation Failure Mode (TR)

The failure of a single component such as a single CPU, a single SMP, or a single communications channel may not cause the full system to become unavailable. It is acceptable for the application executing on a failed CPU or SMP to fail but not for applications executing on other parts of the system to fail.

4.1.5.9 SMP Processor Failure Tolerance (TR)

The SMP may be able to run with one or more computational processors disabled, and to do so with minimal performance degradation. That is, the SMP may be able to tolerate failures through graceful degradation of performance.

4.1.5.10 SMP Memory Failure Tolerance (TR)

The Subcontractor may propose SMPs that are able to run with one or more memory components disabled, and to do so with minimal performance degradation. That is, the SMPs may be able to tolerate failures through graceful degradation of performance.

4.1.5.11 Replacement Parts and Maintenance (MR)

The Subcontractor shall supply hardware and software maintenance for the proposed system for a four year utilization period. Hardware maintenance response time shall be less than four hours from incident report until Subcontractor personnel arrive for repair work. Software maintenance shall include a trouble reporting mechanism and periodic software updates. In addition, the Subcontractor shall provide quick turnaround of software fixes to reported bugs. The proposed system will be installed in a classified area at the Laboratory and so maintenance personnel shall obtain DOE Q clearances.

4.2 SST Software High-Level Requirements

4.2.1 Operating System

4.2.1.1 SMP Base Operating System and License (MR)

The Subcontractor shall provide a standard multiuser POSIX (IEEE 1003.1-1990; FIPS 151-2; IEEE 1003.2 or later) compliant UNIX interactive operating system on each SMP, consisting of a basic kernel that supports system services and multiprocessing applications. Fully supported kernel-level implementation, as defined by the POSIX 1003.4 (or later) working group standard of thread operations in shared address spaces shall also be provided (within six months of standardization or at SST delivery). The operating system shall provide mechanisms to share memory between user processes and to run threads within a single user process on multiple CPUs simultaneously. This shall include provision of right-to-use license for an unlimited number of users, including unlimited concurrent usage, of the operating system, daemons, and associated utilities. The University will accept the Offeror’s self-certification for POSIX compliance.

4.2.1.1.1 X/Open OS Compliance (TR)

The proposed operating system may have the X/Open XPG4 UNIX brand. Software with the functionality of the following X/Open components may be provided: XPG4 C Language ("ISO C"); XPG4 ISAM; FIPS 151-2; XPG4 Commands and Utilities V2; XPG4 Internationalized System Calls and Libraries (Extended); XPG4 X Window System Application Interface (FIPS 158-1). For the XPG4 UNIX brand, the Subcontractor may deliver a copy of the X/Open XPG4 UNIX Brand Certificate with the SST delivery.

4.2.1.1.2 Single System Image (TR)

The SST SMP scalable operating system may present a single system image to users. This may include: single cluster login address; cluster wide load leveling at login and at process creation time; cluster wide file lock management; cluster wide process and POSIX session ID space; virtual memory page sharing between SMPs; and extensions of standard UNIX utilities to the single system image.

4.2.1.1.3 Thread Packages (TR)

The Subcontractor may propose an alternative light-weight threads package tuned for efficiency rather than cross platform portability and provide an implementation of Pthreads as defined by the POSIX 1003.1 standard.

4.2.1.1.4 System Utilities (TR)

The Subcontractor may provide the GNU Utilities, which may at least include: bash, emacs, make; a scripting language such as perl; and lint-like tools for the baseline languages.

4.2.1.1.5 Job Checkpointing (TR)

The Subcontractor may provide an SMP base operating system that can perform process-level checkpointing, at the request of either the job or the job scheduler, and allow that job to continue execution.

4.2.1.2 Cluster Wide Fault Tolerance and Graceful Degradation of Service (TR)

The operating system may have the ability to detect, isolate and manage hardware or software faults in a way that minimizes the impact on overall SMP cluster availability. When SMP cluster (hardware or software) components fail, the SMP cluster software resources may provide degraded system availability. Under most circumstances, it may be possible to take hardware and software components off-line or bring them back on-line without operating system rebooting. The probability that a job will fail (due to hardware or software faults) should be proportional to the amount of resources consumed by the job, not SMP cluster size.

4.2.1.3 Networking Protocols (MR)

The operating system shall support the DoD standard networking protocol suite over the network interfaces described in section 4.1.4.7 of this document. TCP/IP, UDP, NIS, NFS (client and server), RIP, telnet, and ftp protocols shall be supported.

4.2.1.4 Third Party Transfer Support (TR)

The Subcontractor may provide the appropriate networking drivers and protocol support to enable third party transfer of files between the SST and network attached disks controlled by the High Performance Storage System (HPSS).

4.2.1.5 Group Routing (MR)

The Subcontractor shall provide an implementation of “Group Routing,” which segregates network traffic based on (sub)network address and group id. A modified ROUTE table and command that allows (or explicitly disallows) packet routing to specific IP subnets based on group id would satisfy this need.

4.2.1.6 File Systems (TR)

The operating system file system may support a cluster wide file system and individual files of at least one (1.0) TeraByte in size. The file systems may support increased reliability and fast reboots (e.g., reduce the FSCK time via a journal implementation).

4.2.2 Distributed Computing Middleware

4.2.2.1 OSF DCE (MR)

The Subcontractor shall provide the Open Software Foundation (OSF) Distributed Computing Environment (DCE), version 1.1 or later, client software on the proposed cluster of SMPs. This shall include Distributed File System (DFS) client-side distributed filesystem implementation which supports all standard features such as integration with CDS (Cell Directory Service) naming, integration with the DCE security, authentication, and authorization system. Additionally, this shall include fully supported implementation of DCE client-side security system including authentication, authorization controls, and access control lists. Fully supported remote system access programs such as rcp, rlogin, rsh, rexec, telnet, and ftp shall attempt to forward credentials to the remote system; and remote access services such as rcpd, rlogind, rshd, rexecd, telnetd, and ftpd shall accept forwarded credentials from a remote system. If credentials haven't been forwarded, the authentication mechanism of various system services such as login, rlogind, telnetd, and ftpd shall use the DCE security service to authenticate the password which the user presents and shall receive DCE and Kerberos V5 compatible tickets from the DCE security service.

4.2.2.1.1 Distributed File System Server (MR)

The Subcontractor shall provide Open Software Foundation (OSF) Distributed File System Servers, version 1.1 or later. A majority of the file space specified in section 4.1.1.2 shall be exportable with these DFS servers to clients over external networks. The DFS servers shall support file and directory Access Control Lists (ACLs).

4.2.2.1.2 Cluster Wide Service Security (MR)

All cluster wide services including debugging, performance monitoring, event tracing, resource management and control shall be performed in a secure manner using the authentication and authorization capabilities.

4.2.2.1.3 Transarc Encina (MR)

The Subcontractor shall provide a fully supported client-side implementation of the Transarc Encina (version 1.1 or later) distributed transaction system (which is needed to support the HPSS API).

4.2.2.2 Object Request Broker (TR)

The Subcontractor may provide a fully supported implementation of the OMG's (Object Management Group) CORBA (Common Object Request Broker Architecture) version 2.0 (or later) including development tools and libraries for developing object clients and object implementations. CORBA IDL (Interface Definition Language) compilers may build stubs for interfacing with C++.

4.2.3 Cluster Wide Resource Management and Accounting

4.2.3.1 Cluster Wide Resource Management (TR)

The Subcontractor may supply an X11-based graphical user interface for a single point of control resource manager to manage a minimal set of resources which includes number and type of CPUs, per-CPU and aggregate CPU time, wallclock time, memory (high-water allocation), and temporary disk space, as well as more detailed information such as CPU usage by user, network usage by hostname, and memory integral and swap memory integral.

4.2.3.1.1 Cluster Wide Resource Limits (TR)

The Subcontractor may provide a capability to assign and detect/report a soft limit for each supported resource and enforcement of a hard limit for each supported resource (at least the minimal resource set defined in 4.2.3.1).

4.2.3.1.2 Cluster Wide Resource Management API (TR)

The Subcontractor may provide a published API for getting the status for at least the minimal resource set as well as determine the status of overall system resources. The API may also provide interfaces to the following capabilities: selectively preempt or revoke critical resources from a job; set default provisions for abort, suspend, or do nothing for invocation at hard or soft limits boundaries; if the default provision is to suspend or abort, then a job priority ranking interface may be provided by the system so that the distributed resource manager can inform the system as to which job to affect in the default.

4.2.3.2 Cluster Wide Job Accounting (TR)

The Subcontractor may provide job accounting, with API interface, where data for all threads and processes of a job are combined to provide an aggregate job accounting record, and individual thread and process records are identified as being related to a job, for at least the minimal resource set. The accounting information may also contain UNIX UID information. The data returned may be an up-to-date snapshot of resource usage as of the time of the call to the API routine.

4.2.3.3 Accounting for System (Root) Usage (TR)

Resources used by root processes that are not otherwise considered to be part of another job and by the operating system itself may be accountable. In addition, resources not used may be accountable. Accounting for these resources may be accomplished through the same interfaces as are provided for all other jobs. Accounting for operating system resource usage may be done through a pseudo-job known as the “kernel” job. Accounting for root processes’ resource usage may be done through one or more pseudo-jobs known as “system” jobs. Finally, accounting for unused resources may be done through a pseudo-job known as the “idle” job.

4.2.3.4 Cluster Wide Job Management (MR)

A job shall be a cluster wide abstraction similar to a POSIX session, with certain characteristics and attributes. Commands shall be available to manipulate a job as a single entity (including kill, modify, query characteristics, and query state). The characteristics and attributes required for each session type are as follows: 1) interactive session: an interactive session shall include all cluster wide processes executed as a child (whether direct or indirect through other processes) of a login shell and shall include the login shell process as well. Normally, the login shell process shall exist in a process chain as follows: init, inetd, [telnetd | rlogind | xterm | cron], then shell. 2) batch session: a batch session shall include all cluster wide processes executed as a child (whether direct or indirect through other processes) of a shell process executed as a child process of a batch system shepherd process, and shall include the batch system shepherd process as well. 3) ftp session: an ftp session shall include an ftpd and all its child processes. 4) kernel session: all processes with a pid of 0. 5) idle session: this session does not necessarily actually consist of identifiable processes. It is a pseudo-session used to report the lack of use of resources. 6) system session: all processes owned by root that are not a part of any other session.

4.2.3.5 Cluster Wide Job Scheduling (MR)

The Subcontractor shall provide a capability so that a user can submit, either interactively or through the batch environment, a job that spans any subset of user accessible processors, and to schedule multi-thread, multi-process, message-passing jobs using configurable load levels. These configurable load levels shall be defined on CPU load, available real memory, paging rate (if applicable), available swap and temporary disk space.

4.2.3.5.1 Cluster and SMP Gang Scheduling (TR)

The Subcontractor may provide a capability to gang schedule threads and processes from a single user job within an SMP and across SMPs. That is, when a user job is scheduled to run, the gang scheduler may contemporaneously allocate to CPUs all the threads and processes within that job (either within an SMP or within the cluster of SMPs). This scheduling capability may control all threads and processes within the SMP cluster environment.

4.2.3.5.2 Cluster Wide Job Scheduling Interface (TR)

The Subcontractor may provide an X11-based graphical user interface using an X window display which displays machine loads and systems information (including as a minimum the items described in 4.2.3.5) . This interface may also provide historical usage, job tracking from submission to deletion (note that deletion is not the same as end of execution), capability to control distribution of resources between interactive and batch requests, ability to tailor the scheduling policy to specific site requirements, and the ability to adjust cluster-wide load balance by checkpoint/migration/restart of single process jobs. In addition, job status shall be retained at least 24 hours after termination. All of the job scheduling functions underlying this X11-based user interface may also be provided in a published API.

4.2.4 Cluster System Administration Tools

4.2.4.1 Single Point for Cluster System Administration (MR)

The Subcontractor shall provide a set of facilities to administer the SST SMP cluster as a single entity. In particular, the Subcontractor shall provide fully supported implementation of a single-point system administration tool to handle: file system mounts; SMP booting, where appropriate; SMP status, where appropriate; SMP self-consistency checks of system configuration parameters; software installation; resource administration; SMP shutdown/restart; system patch installation; login control (provide capability to restrict login access to certain processors, and cluster-wide monitoring of failed login attempts by an individual); and system back-ups, including ability to dump multiple volumes of tapes without operator intervention. The Subcontractor shall provide a fully supported implementation of mechanisms for detecting and reporting failures of critical resources, including processors, network paths, and disks. The diagnostic routines shall be capable of isolating hardware problems down to the FRU level in both the system and its peripheral equipment.

4.2.4.2 Cluster System Debugging and Performance Analysis (TR)

The Subcontractor may provide a set of facilities to analyze SST system performance and make tuning modifications. In particular, the Subcontractor may provide fully supported implementation of a single-point system tuning tool to dynamically monitor and modify the following system attributes: processor status; key resources: system CPU usage, memory usage, page faults; run queues per SMP; scheduling priority of each process and each thread within a process; and current system configuration. The tuning parameter changes may take affect without requiring an operating system reboot.

4.2.4.3 Centralized Repository (TR)

The Subcontractor may provide a centralized resource data repository, keeping track of the state of all system resources, their current usage policies, and a system error log.

4.2.4.4 User Maintenance (TR)

The Subcontractor may provide a tool for managing user administration, including some means of integrating the namespace manager and the authentication server in order to facilitate adding, removing, and modifying users. In addition, the Subcontractor may provide a tool for managing groups, including initial creation of groups, modification of groups, and user membership in groups.

4.2.5 Parallelizing Compilers/Translators

4.2.5.1 Baseline Languages (MR)

The Subcontractor shall provide fully supported implementations of Fortran90 (ISO 1539-1), C (ANSI/ISO 9899-1992;FIPS 160 or later), and C++. The Subcontractor shall provide a version of C++ that is standard compliant within one year after standardization. The Subcontractor is encouraged to adhere to the current proposed standard. Fortran90, C, and C++ shall be referred to as the baseline languages in this section. Also, an assembler shall be provided. The Subcontractor shall provide the fully supported capability to build programs from a mixture of the baseline languages (i.e., inter-language subprocedure invocation must be supported). The Fortran90 and C compilers shall support parallelism through directives or language constructs. As an optimization feature, the Fortran90 and C compilers shall perform automatic parallelization.

4.2.5.1.1 Common Preprocessor for Baseline Languages (TR)

The Subcontractor may provide the capability of preprocessing ANSI C preprocessor directives in programs written in any of the baseline languages.

4.2.5.1.2 Base Language Interprocedural Analysis (TR)

The Subcontractor may provide mechanisms to perform basic interprocedural analysis (e.g., variable cross-reference listing, COMMON block analysis, use/def analysis) for programs written in the baseline languages.

4.2.5.1.3 Baseline Language Compiler Generated Listings (TR)

The Subcontractor may provide baseline language compiler option(s) to produce source code listings that include information such as pseudo-assembly-language listings, optimizations and inhibitors, and variable types.

4.2.5.1.4 C++ Functionality (TR)

The Subcontractor may provide an implementation of the C++ standard template library and exception handling.

4.2.5.1.5 Cray Pointer Functionality (TR)

The Subcontractor may provide Cray style pointers implemented in an ANSI X3.9-1977 Fortran compliant compiler.

4.2.5.2 High Performance Fortran (TR)

The Subcontractor may propose a system with a fully supported implementation of HPF Version 1.1 or HPF Version 2.0. Additional extended HPF features to support indirect distribution, asynchronous I/O, C-interface, and distribution of arrays of derived types are of specific interest. Additional platform-specific extensions are allowed.

4.2.6 Debugging and Tuning Tools

4.2.6.1 Debugger for Cluster Wide Applications (MR)

The Subcontractor shall provide an interactive debugger with an X11-based graphical user interface enabling single-point of control (multiple debugger invocations to control individual processes are not acceptable) that can debug cluster-wide applications with multiple parallel programming paradigms (e.g., message passing, thread and process parallelism). The parallel debugger shall function at the initial source level (before any preprocessing) for programs developed with inter-mixed baseline languages. A command line interface shall be available for sequential and parallel programs. The capability of attaching/detaching the debugger to/from an executing (serial or parallel) program and modifying program state and continuing execution shall be provided. If the code was not compiled for debugging, it is understood that access to source-level information may be limited. For message passing codes the debugger shall display the status of message queues, such as number of pending messages and associated length, source, and sink at a breakpoint. Debugger functionality shall include, but is not limited to: control of processes and threads (start/stop, breakpoints, and single-step into/over subprocedure invocations); examination of program state (stack tracebacks, contents of variables, array sections, aggregates, and blocks of memory, current states, registers, and source locations of processes); and modification of program state (changes to contents of variables, aggregates, and blocks of memory).

4.2.6.1.1 Visual Representation of Data (TR)

The Subcontractor may provide a parallel debugger capable of displaying multiple visual representations of values in a matrix or 2-D array section (e.g., bitmap showing elements exceeding a threshold value, colormap, surface map, contour map) with zoom and pan capability for the visual displays to facilitate scaling and display of large arrays. This functionality may be provided for all baseline languages. The Subcontractor may provide a conditional data filtering capability for large data sets integrated with data display functions, both textual and graphic. All visual capability may be invoked directly from the debugger.

4.2.6.1.2 Fast Conditional Data Watchpoints and Breakpoints (TR)

The Subcontractor provided debugger may support fast conditional data watchpoints and breakpoints in all the baseline languages. That is, an implementation for memory or source code breakpoints which does not degrade the application runtime within the debugger more than a factor of four for watch points (1.5 for breakpoints) relative to the runtime without conditional data watchpoints (or source code breakpoints) set.

4.2.6.1.3 Memory Leak Debugging (TR)

The Subcontractor may provide the capability of reporting memory access errors and pointing to the offending source line in C and C++. Memory access errors reported may include: accessing/freeing beyond allocated block; accessing/freeing unallocated blocks; memory leaks (accumulated memory chunks from malloc calls that can no longer be accessed or freed); and uninitialized memory read/write.

4.2.6.1.4 Debugging Optimized Applications (TR)

The parallel debugger may, in the presence of code optimization, provide a fully supported mechanism for reporting at least minimal information on program state (stack traceback, access to variables that have not been eliminated) and some degree of functionality such as breakpoints where possible (including at labeled basic blocks), single-stepping at some level, and stepping over subroutines.

4.2.6.1.5 Debugger Expression Evaluator (TR)

The parallel debugger may have an evaluator capable of calculating the results of simple expressions (in some scripting-like language) such as values of conditionals, indirect array references, etc. It is also desired that the evaluator handle the supported languages. This might be a language interpreter, but for the purposes of user code to be executed at breakpoints, or watchpoints, some form of compiled code is more desirable to make impact on execution smaller.

4.2.6.1.6 Parallel Debugger Barrier-Points (TR)

The parallel debugger may have an expanded breakpoint functionality for control of parallel processes by setting a “barrier-point.” With a barrier point, the process may be held until all processes reach the same point, not responding to “start” commands until the barrier point is satisfied, or released.

4.2.6.1.7 Post-Mortem Debugging (TR)

The debugger may have a fully supported implementation of some mechanism for invoking the debugger for examining the final state of a program that failed ("postmortem debugging"). Facilities for modifying program state and/or continuing execution need not be available in this mode. If the code was not compiled for debugging, it is understood that access to source-level information may be limited.

4.2.6.1.8 Symbol Table (TR)

The time to initialize the debugger on an application with a 15 MB symbol table may be less than a minute longer than the time to initialize the debugger on the same number of processors, but with no symbol table.

4.2.6.1.9 Data Aggregation (TR)

The parallel debugger may have a capability of accumulating the local values of variables that are replicated across multiple threads / processes, and presenting a condensed summary within a single window. In addition, where distributed arrays are supported by the programming model, capability of gathering the elements of a distributed 2-D array and presenting them in a single tabular / visualization.

4.2.6.2 Stack Traceback (TR)

The Subcontractor may provide runtime support for stack traceback error reporting. Critical information may be generated to stderr upon interruption of a process or thread involving any trap for which the user program has not defined a handler. The information may include a source-level stack traceback (indicating the approximate location of the process or thread in terms of source routine and line number) and an indication of the interrupt type.

4.2.6.3 Lightweight Corefile API (TR)

The Subcontractor may provide the standard lightweight corefile API, defined by the Parallel Tools Consortium, to trigger generation of aggregate traceback data like that described in 4.2.6.2 (may produce a platform-specific format). The specific format and command-line and graphical browsers for the lightweight corefile facility defined by the Parallel Tools Consortium are also of interest.

4.2.6.4 Profiling Tools for Cluster Applications (MR)

The Subcontractor shall provide tools for profiling CPU time distribution from all processes or threads in a parallel program, at the levels of subprocedures and coarse blocks (e.g., large loops). The tools shall include a capability for restricting the amount of profiling data collected to certain portions of the source code (e.g., a specific subset of procedures), through the use of compiler directives, API or command-line switches.

4.2.6.5 Event Tracing Tools for Cluster Applications (MR)

The Subcontractor shall provide event tracing tools for cluster wide applications. Distributed mechanisms for generating event records from all process and threads in the parallel program shall include timestamp and event type designator and shall be formatted in a self-defining data format. This functionality must be provided for all baseline languages. The event tracing tool API shall dynamically activate and deactivate event monitoring during execution from both within and outside a process.

4.2.6.5.1 Message-Passing Event Tracing (TR)
The Subcontractor may provide a fully supported implementation of some mechanism for tracing message sends, receives, and synchronizations, including non-blocking messages, for all message-passing libraries supported on the platform.

4.2.6.6 Performance Statistics Tools for Cluster Applications (MR)
The Subcontractor shall provide performance statistics tools, whereby performance measures obtained for individual threads or processes are reported and summarized for the cluster wide application. There shall be a mechanism for capturing the statistics and storing them for later analysis/viewing. Statistics of interest include: instruction issues by type; branch prediction accuracy; Translation Lookaside Buffer (TLB) and cache misses; lock or critical section usage; page faults; I/O and communications (in terms of bytes sent/received). The measures shall also include a summary of memory usage (actual allocations, not memory references) and a mechanism to relate that back to baseline language source code.

4.2.6.7 Cluster Wide Application Development Tool API (TR)
The Subcontractor may provide a published API (possibly platform-specific) for dynamically activating and deactivating profiling, event tracing, and performance statistics during execution.

4.2.6.8 Cluster Wide Application Development Tool GUI (MR)
The Subcontractor shall provide graphical as well as textual displays of profiling, event tracing and performance statistics tools output.

4.2.6.9 Timer API (TR)
The Subcontractor may provide an implementation of the API defined by the Parallel Tools Consortium for interval wallclock and for interval CPU timers local to a thread/process. These timer interfaces may access the best available wallclock timer on the platform, in terms of accuracy and non-intrusiveness.

4.2.7 Applications Building

4.2.7.1 Linker and Library Building Utility (MR)
The Subcontractor shall provide an application linker with the capability to link object and library modules into an executable binary. In addition the linker shall be capable of re-linking selected portions of an application (i.e., replace specific objects within the binary). The Subcontractor shall include a facility to build and incrementally update libraries of object modules.

4.2.7.2 Make Utility (MR)
The Subcontractor shall provide a make utility.

4.2.7.3 Parallel Make (TR)
The Subcontractor may provide a UNIX make facility that can utilize parallelism in performing the tasks in a makefile.

4.2.7.4 Source Code Management (TR)

The Subcontractor may provide a set of tools for the management of source code in a multiple programmer project environment (e.g., SCCS, USM, RCS, CVS).

4.2.7.5 Dynamic Processor Allocation (TR)

The Subcontractor may provide the capability of running an application on varying numbers of processors and/or threads without recompilation or relinking.

4.2.8 Application Programming Interfaces

4.2.8.1 Optimized Message-Passing Interface (MPI) Library (MR)

The Subcontractor shall provide a fully supported implementation of the current MPI standard, as defined by the most recent specification of the MPI forum. The MPI library shall operate transparently on all required interconnects, including shared memory.

4.2.8.2 MPI2 Library (TR)

The Subcontractor may make available an implementation of the evolving MPI2 standard (specifically dynamic spawning and single-sided communication capabilities) within one year of standard finalization.

4.2.8.3 Visual Display of Message Traffic (TR)

The Subcontractor may make available an implementation of a visual API to display message traffic and program state in a time-space diagram (like that generated by XPVM or AIMS).

4.2.8.4 Optimized Parallel Virtual Machine (PVM) Library (TR)

The Subcontractor may provide an implementation of PVM within six months of latest release from the Oak Ridge National Laboratory (ORNL).

4.2.8.5 Parallel I/O API (TR)

The Subcontractor may provide a fully supported implementation of a published API supporting four kinds of concurrent file I/O as described below. All threads and processes in a parallel program may be capable of performing the four operations described below although there may be variation in performance from one thread or process to another. It is the application's responsibility to ensure that all participating threads and processes open the logical file in the same mode.

1. Sequential read: all participating threads and processes read from a logically shared file using a shared file pointer. Each record may be read just once.

2. Parallel read: all participating threads and processes read from a logically shared file using independent file pointers. Thus, each process reads each record.

3. Sequentialized write: all participating threads and processes write to a logically shared file using a shared file pointer. Records are atomic and cannot be overwritten, but they may be merged into the shared output file in any order.

4. Direct access read/write: each participating thread or process can read or write any specified record location of a logically shared file. Updates are not guaranteed to take effect until the file is closed, but if the implementation makes use of shared buffers, records shall be atomic (i.e., not overwritten by other processes).

4.2.8.5.1 Parallel I/O Buffer Attributes (TR)

The supplied parallel I/O library may have user configurable buffer lengths and all buffers associated with a job may be automatically flushed upon completion or failure of the job. It may also have an API for application invoked flushing of individual buffers.

4.2.8.5.2 Parallel I/O Sequential Write (TR)

Output from a cluster wide parallel job directed to stdout/stderr may be atomic on a line by line basis and be prefixed with a label indicating which thread or process performed the write.

4.2.8.5.3 Asynchronous Parallel I/O (TR)

The supplied parallel I/O library may also support asynchronous read/write operations in all the parallel I/O modes.

4.2.8.5.4 High-Level Parallel I/O API (TR)

The supplied parallel I/O library may also support an interface to a self-describing data-format, such as netCDF or HDF.

4.2.8.6 Graphical User Interface API (MR)

The Subcontractor shall provide the standard X11R6 and Motif 1.2, or current versions, applications, servers and API libraries.

4.2.8.7 Visualization API (TR)

The Subcontractor may provide OpenGL 1.0, or current version, and the Programmers Hierarchical Interactive Graphics System (PHIGS), ISO/IEC 9592-1:1988, including ISO/IEC 9592-4:1992 Plus Lumierre (lighting) and Surfaces (PHIGS PLUS), or current version.

4.2.8.8 Math Libraries (TR)

The Subcontractor may supply highly optimized mathematical libraries, callable from all baseline languages, for serial (single processor) SMP and cluster wide applications.

4.2.9 Operating System Security (TR)

The Subcontractor may provide security functionality where access to the system may be controlled by identifying and authorizing the user or by checking the validity of forwarded credentials. All users may be authenticated before access is permitted.

Successive logon attempts may be controlled by denying access after multiple (maximum of 5) unsuccessful logon attempts with the same user ID.

4.2.9.1 Login Information (TR)

Users may be notified upon successful login of the following information: date and time of last successful login; and where the operating system provides the capability, number of unsuccessful attempts.

4.2.9.2 Audit Capability (MR)

A record of each user login and logoff shall be maintained. In addition, the following information shall be maintained as an audit record: use of authentication changing procedures; unsuccessful logon attempts; and blocking of a user ID and the reason for the blocking.

4.2.10 Compliance with DOE Security Mandates (MR)

DOE Security Orders have changed over time. From time to time these mandates cause LANL and LLNL to fix bugs or implement security features in vendor operating systems and utilities. We require, for computer security purposes only on this contract, that the Subcontractor either provide to the University the operating system and utilities source code on a demand basis or make needed enhancements or bug fixes as required.

4.2.11 Cluster Environment On-Line Document (TR)

The Subcontractor may supply hardcopy and on-line documentation for all major hardware and software subsystems. The on-line documentation may be readable by all users utilizing an X11-based graphical user interface and standard terminal interfaces.

4.2.12 SST Applications Development Support (MR)

The Subcontractor shall supply at least two on-site analysts to provide expertise to the University code development teams in the areas of software development tools, parallel applications libraries and applications performance.

End of Section 4

5.0 ID High-Level Technical Requirements

The ASCI program requires early access to stable code development platforms for the rapid development of SBSS applications for the Sustained Stewardship TeraFLOP/s (SST) system. It is imperative that the Initial Delivery (ID) code development environment present the same hardware and software model to code developers that will be available on the SST. In particular, hardware issues such as memory hierarchy and message passing support must present the same issues to application performance that will be experienced in the SST. As an example, if the high cache utilization is required for applications performance on the SST then the ID must present the same performance challenges and opportunities for optimization.

It is envisioned that the Initial Delivery (ID) come from the subcontractor's current (or very near term) product offering and provide, to the maximum extent possible, a code development environment congruent with that of the SST.

In specifying the ID system the University was motivated by two factors: 1) provide a capability beyond (approximately double) what is currently available at either site; 2) provide a hardware and software code development environment that closely resembles the SST. This ID system will be delivered either to LANL or LLNL with an optional second delivery.

Due to the classified and unclassified ASCI programmatic requirements this system must function in the classified (RED) and unclassified (BLACK) networking partitions. For the ID system, the thinking is that a stable code development environment on both the RED and BLACK partitions is the most appealing solution. This implies that the ID system would need to be split between the RED and BLACK partitions. This solution would allow for permanent, local disks on each partition divided in some way from the total requirement.

However, an alternate, less attractive solution would be to frequently (on the order of once a day) swing the resource between the two partitions. For this scenario, we require that the transition time between RED and BLACK (and vice-versa) be less than thirty (30) minutes. When the system is on the RED (or BLACK) partition, users on the BLACK (or RED) partition still require access to the BLACK (or RED) file system partition to edit their files and rebuild binaries. These requirements imply (for the hardware) that no CPU or SMP node local disks may be employed. In addition, the Subcontractor might need to supply two separate disk systems with powerful file servers which can be detached and reattached on very short time scales. The aggregation of these two I/O partitions would be used to compute the total disk provided (i.e., the disk requirement does not need to be doubled).

The specific mandatory hardware and software requirements the ID system must meet are delineated in section 5.1, Initial Delivery Hardware Requirements, and section 5.2, Initial Delivery Software Requirements with (MR) designations.

In addition to the mandatory hardware and software requirements, the Subcontractor may deliver any Target Requirements (TR) for the ID, and any additional features consistent with the objectives of this project and Subcontractor's Research and Development Plan, which the Subcontractor believes will be of benefit to the project.

5.1 Initial Delivery Hardware Requirements

5.1.1 Hierarchical Hardware Model

5.1.1.1 ID SMP Scalable Cluster (MR)
The Subcontractor shall provide an Initial Delivery (ID) system that implements all aspects of the hierarchical distributed shared memory programming model. The hierarchy consists of processor registers, cache, shared memory and communications. The hierarchy shall present the same performance aspects as the SST SMP scalable cluster. All cluster components need not present all of the memory hierarchy components. All cluster components shall be connected via a scalable network technology. The scalable cluster network shall support high bandwidth, low latency message passing. Within SMP cluster components the CPUs shall be connected via a high speed, extremely low latency mechanism to the set of hierarchical memory components. The cache may be hierarchical. If there are multiple SMP caches, they shall be kept coherent automatically by the hardware. The memory may be a Non-Uniform Memory Access (NUMA) architecture.

5.1.1.2 CPU Performance Model (TR)
The ID system may present similar general CPU performance characteristics to ASCI applications as the SST. The University is aware that the specifics (e.g., number of FP operations issued per clock, instruction pipeline depth, off chip bandwidth, branch prediction and dynamic execution) between the two systems may differ.

5.1.1.3 Hierarchical Memory Model (TR)
The ID system may present a similar general hardware hierarchical memory model to ASCI applications as the SST does. The University is aware that the specifics (e.g., cache sizes, bandwidths and latencies, non-unit stride access performance and ratio of remote to local memory references) between the two systems may differ.

5.1.1.4 Message Passing Model (TR)
The ID system may present a similar general message passing and remote memory reference hardware interface to ASCI applications as the SST does. The University is aware that the specifics (e.g., bandwidths and latencies, remote reference granularity and software overhead) between the two systems may differ.

5.1.1.5 Parallel I/O Model (TR)
The ID system may present a similar general parallel I/O hardware model to ASCI applications as the SST does. The University is aware that the specifics (e.g., number of file servers, I/O channel bandwidths and latencies and software overhead) between the two systems may differ.

5.1.2 ID Performance Characteristics

5.1.2.1 ID Single CPU Performance (TR)

The minimum single CPU performance, as measured by peak floating point operations issued per second, may be at least two hundred fifty million 64-bit floating point operations per second (250 MFLOP/s).

5.1.2.2 ID SMP Cluster Performance (MR)

The ID SMP cluster performance, as measured by aggregate peak floating point performance, shall be at least one hundred billion 64-bit floating point operations per second (100 GFLOP/s).

5.1.2.3 ID System Component Scaling (TR)

In order to provide the maximum flexibility to the Subcontractor in meeting the goals of the ASCI project the exact configuration of the ID SMP scalable cluster is not specified. Rather the ID configuration is given in terms of lower bounds on component attributes. The ID SMP scalable cluster configuration may meet or exceed the following parameters:

- Memory Size ≥ 50 GB
- Disk Space ≥ 2.5 TB
- Cache Bandwidth ≥ 400 GB/s
- Memory Bandwidth ≥ 100 GB/s
- Intra-Cluster Network Bi-Section Bandwidth ≥ 50 Gb/s
- System Peak Disk I/O Bandwidth ≥ 3 GB/s

5.1.2.4 ID SMP Cluster Message Passing Performance (TR)

The ID SMP may be interconnected with at least one high speed, low latency network. Each SMP may have at least one connection link to this network capable of sustaining at least 0.1 Gb/s per peak GFLOP/s of processing power of that SMP. The cluster interconnect latency, as measured by sending a minimum length MPI message from user program memory on one processor in the SMP cluster to user program memory on any other processor in the cluster and receiving back an acknowledgment divided by two (standard MPI user space ping-pong test), may be less than 10 micro-seconds ($1x10^{-5}$ seconds).

Example: If the Subcontractor proposes 13 SMP's each with an aggregate of 8.0 GFLOP/s peak, then the cluster interconnect link bandwidth may be at least 0.1*8 Gb/s = 0.8 Gb/s.

5.1.2.5 Sustained ID Serial and Parallel I/O Performance (TR)

The minimum sustained transfer rate for reading or writing of a single 1.0 GB file to or from a single logical file system anywhere in the active I/O partition may be at least 10 MB/s. The minimum sustained parallel I/O transfer rate for reading or writing of a parallel 10.0 GB file to a parallel file system in the active I/O partition may be at least 0.002 Byte/s per Peak FLOP/s.

Example: For a 100 GFLOP/s peak ID system, requirement 5.1.2.3 specifies that the system shall have at least 2.5 TB disk and 3.0 GB/s system peak I/O bandwidth. The sustained parallel I/O bandwidth to a single application running on the cluster may be at least 0.002*100 = 0.2 GB/s. This would allow an application write a 50 GB restart dump in a little under 4.2 minutes.

5.1.2.6 ID SMP Cluster External Networking (TR)

The ID system may include a minimum of four (4) FDDI interfaces (dual attached, 100 Mb/s) . The ID system may include at least 8 HIPPI connections (800 Mb/s). These external networking connections may be split between the RED and BLACK partitions. Each of these connections may be capable of interoperating with existing LANL and LLNL systems and network gateways.

5.1.3 RED/BLACK Resource Usage Model

5.1.3.1 RED/BLACK Partitions (MR)

The ID system resources shall be available to ASCI code developers on both the RED and BLACK partitions at least part of the time. Code developers require constant access to the file systems on both partitions and the ability to compile, make libraries, link and load (but not run or debug) binaries. Two general approaches to this problem are specified. The system provided shall meet the requirements of one of these two approaches. An alternate solution that meets the requirements stated above may also be considered at the sole discretion of the University.

5.1.3.1.1 RED/BLACK Resource Split (MO)

The scalable SMP cluster shall be divided in two (not necessarily equal) separate, stand-alone clusters. One cluster shall be placed in the RED network partition and one cluster shall be placed in the BLACK partition for ASCI code development. The CPU, memory, disk, scalable cluster interconnect and external network resources shall be divided between the two environments. Disks local to the SMPs are acceptable with this option.

5.1.3.1.2 RED/BLACK Partition Switching (MO)

The external network and aggregate disk resources of the scalable SMP cluster shall be divided in two (not necessarily equal) separate, I/O subsystems that can be attached and detached to the SMP cluster (one RED and one BLACK). The SMP cluster shall boot from either of these I/O subsystems. When the SMP cluster is attached to the RED (or BLACK) I/O subsystem and external network the files contained on the BLACK (or RED) subsystem shall be available to users over BLACK (or RED) external network connections. Users shall be able to edit, compile and load their programs utilizing this BLACK (or RED) resource. The system shall be able to switch between the RED and BLACK I/O partition and external network in less than thirty (30) minutes. There shall be no user-writeable non-volatile memory (for any other than privileged users) in the proposed system except in the detachable, independent RED/BLACK I/O subsystems.

5.1.3.2 ID RAID Arrays (TR)

All disk space provided in the RED/BLACK partitions may be RAID 3 or RAID 5 arrays. The RAID units may have high availability characteristics. These include redundant failover power supplies and fans, hot swapable disks, ability to run in degraded mode (one disk/RAID string failure) and rebuild a replaced disk on the fly with minimal performance impact.

5.1.3.3 Cluster I/O Upgradeability (TR)

The disk capacity may be field upgradeable to twice the Subcontractor proposal amount.

5.1.4 Reliability, Availability, Serviceability and Maintenance (TR)

The Subcontractor may provide the same reliability, availability, serviceability and maintenance for the ID system as for the SST system as described for the Target Requirements in section 4.1.5.

5.1.4.1 Replacement Parts and Maintenance (MR)

The Subcontractor shall supply hardware and software maintenance for the proposed system. Hardware maintenance response time shall be less than four hours from incident report until Subcontractor personnel arrive for repair work. Software maintenance shall include a trouble reporting mechanism and periodic software updates. In addition, the Subcontractor shall provide quick turnaround of software fixes to reported bugs. The proposed system will be installed in a classified area at the Laboratory and so maintenance personnel shall obtain DOE Q clearances.

5.2 Initial Delivery Software Requirements

The general philosophy for the ID software requirements is to pose as few requirements as is feasible and reference all of the SST software target requirements (see section 5.2.9). This is done for the sake of brevity and does not indicate a lack of perceived value by the University in the software environment on the ID system.

5.2.1 Operating System

5.2.1.1 SMP Base Operating System and License (MR)

The Subcontractor shall provide a standard multiuser POSIX (IEEE 1003.1-1990; FIPS 151-2; IEEE 1003.2 or later) UNIX interactive operating system on each SMP, consisting of a basic kernel that supports system services and multiprocessing applications. Fully supported kernel-level implementation, within six months of the finalization of the POSIX 1003.4 working group standard of thread operations in shared address spaces must also be provided. The operating system shall provide mechanisms to share memory between user processes and to run threads within a single user process on multiple CPUs simultaneously. This shall include provision of right-to-use license for an unlimited number of users, including unlimited concurrent usage, of the operating system, daemons, and associated utilities. The University will accept the Offeror’s self-certification for POSIX compliance.

5.2.1.2 Networking Protocols (MR)

The operating system shall support the DoD standard networking protocol suite operating over the network interfaces described elsewhere in this document. In particular, the TCP/IP, UDP, NIS, NFS (client and server), RIP, telnet, and ftp protocols shall be supported.

5.2.1.3 Third Party Transfers (TR)

The Subcontractor may provide a driver that supports third party transfers of files between the ID and HIPPI network attached disks controlled by NSL UniTree version 2.1, or later. This implies an operating system driver capable of IPI-3 over HIPPI with National Storage Laboratory third party extensions (see LLNL Report UCRL-ID-123184).

5.2.1.4 Group Routing (MR)

The Subcontractor shall provide an implementation of “Group Routing,” which segregates network traffic based on (sub)network address and group ID. A modified ROUTE table and command that allows (or explicitly disallows) packet routing to specific IP subnets based on group ID would satisfy this need.

5.2.2 Distributed Computing Environment (MR)

The Subcontractor shall provide the Open Software Foundation (OSF) Distributed Computing Environment (DCE), version 1.1 or later, client software on the proposed cluster of SMPs. This shall include Distributed File System (DFS) client-side distributed filesystem implementation which supports all standard features such as integration with CDS naming, integration with the DCE security, authentication, and authorization system. Additionally, this shall include fully supported implementation of DCE client-side security system including authentication, authorization controls, and access control lists. Fully supported remote system access programs such as rcp, rlogin, rsh, rexec, telnet, and ftp shall attempt to forward credentials to the remote system; and remote access services such as rcpd, rlogind, rshd, rexecd, telnetd, and ftpd shall accept forwarded credentials from a remote system. If credentials haven't been forwarded, the authentication mechanism of various system services such as login, rlogind, telnetd, and ftpd shall use the DCE security service to authentication the password which the user presents and shall receive DCE and Kerberos V5 compatible tickets from the DCE security service.

5.2.2.1 Distributed File System Server (MR)

The Subcontractor shall provide Open Software Foundation (OSF) Distributed File System Servers, version 1.1 or later. A majority of the file space specified in section 5.1.2.3 shall be exportable with these DFS servers to clients over external network. The DFS servers shall support file and directory Access Control Lists (ACLs).

5.2.2.2 Transarc Encina (MR)

The Subcontractor shall provide fully supported client-side implementation of the Transarc Encina (version 1.1 or later) distributed transaction system (which is needed to support the HPSS API).

5.2.3 Cluster Wide Resource Management and Accounting (TR)

The Subcontractor may provide a network batch queuing system (e.g., NQS, LSF) that allows for the execution of batch jobs within the cluster environment. The Subcontractor may provide a standard UNIX session based accounting mechanism that accurately accounts for all activity in the system on an end of process basis.

5.2.4 Cluster System Administration Tools (TR)

The Subcontractor may provide a set of facilities to administer the ID SMP cluster as a single entity. Such facilities may include, but are not limited to: file system mounts; SMP booting; SMP status, where appropriate; SMP consistency checks; software installation; resource administration; SMP shutdown/restart; system patch installation; login control; and system back-ups. In addition, the Subcontractor may provide a set of facilities to analyze ID system performance and make tuning modifications.

5.2.5 Parallelizing Compilers/Translators

5.2.5.1 Baseline Languages (MR)

The Subcontractor shall provide Fortran90 (ANSI X3.198-9 or later), C (ANSI/ISO 9899-1992; FIPS 160 or later), and C++ compilers. These languages shall be referred to as the baseline languages in this document. Also, an assembler shall be provided. The Subcontractor shall provide the fully supported capability to build programs from a mixture of the baseline languages (i.e., inter-language subprocedure invocation must be supported).

5.2.5.1.1 Cray Pointer Functionality (TR)

The Subcontractor may provide Cray style pointers implemented in an ANSI X3.9-1977 Fortran compliant compiler.

5.2.5.2 Hierarchical Programming Model (TR)

The ID programming environment may present a similar general hierarchical programming model to the programmer as that anticipated for the SST. This model may include the notion of hierarchical amalgamation of memories and latencies (e.g., registers, cache, SMP local memory and SMP remote memory) and code development techniques (e.g., local memory organization techniques, threads, processes, message passing) to utilize them efficiently. The University is aware that the specifics (e.g., compiler optimization features, support for multiple parallel programming paradigms in a single program, support for remote memory reference, hardware features) between the two programming environments may differ.

5.2.6 Debugging and Tuning Tools

5.2.6.1 Debugger for Parallel Applications (MR)

The Subcontractor shall provide an interactive debugger with an X11-based graphical user interface for debugging both sequential and parallel programs for all the baseline languages. The debugger shall report information at the level of program source code (before preprocessing) for all baseline languages, including support for mixed-language programs. Functionality shall include, but is not limited to: control of processes and threads (start/stop, breakpoints, and single-step into/over subprocedure invocations); examination of program state (stack tracebacks, contents of variables, aggregates, and blocks of memory, current states, registers, and source locations of processes); and modification of program state (changes to contents of variables, aggregates, and blocks of memory).

5.2.7 Applications Building

5.2.7.1 Linker and Library Building Utility (MR)

The Subcontractor shall provide an application linker with the capability to link object and library modules into a executable binary. The Subcontractor shall include a facility to build and incrementally update libraries of object modules.

5.2.7.2 Make Utility (MR)

The Subcontractor shall provide a make utility.

5.2.8 Application Programming Interfaces

5.2.8.1 Optimized Message-Passing Interface (MPI) Library (MR)

The Subcontractor shall provide a fully supported implementation of the current MPI standard, as defined by the most recent specification of the MPI forum. The MPI library shall operate transparently on all required interconnects, including shared memory.

5.2.8.2 Optimized Parallel Virtual Machine (PVM) Library (TR)

The Subcontractor may provide implementation of PVM within six months of latest release from Oak Ridge National Laboratory (ORNL).

5.2.8.3 Parallel I/O API (TR)

The Subcontractor shall provide a fully supported implementation of a published API supporting four kinds of concurrent file I/O as listed below. All threads and processes in a parallel program shall be capable of performing the four operations described below although there may be variation in performance from one thread or process to another. It is the programmer's responsibility to ensure that all participating threads and processes open the logical file in the same mode.

1. Sequential read: all participating threads and processes read from a logically shared file using a shared file pointer. Each record is read just once.

2. Parallel read: all participating threads and processes read from a logically shared file using independent file pointers. Thus, each process reads each record.

3. Sequentialized write: all participating threads and processes write to a logically shared file using a shared file pointer. Records are atomic and cannot be overwritten, but they may be merged into the shared output file in any order.

4. Direct access read/write: each participating thread or process can read or write any specified record location of a logically shared file. Updates are not guaranteed to take effect until the file is closed, but if the implementation makes use of shared buffers, records shall be atomic (i.e., not overwritten by other processes).

5.2.8.3.1 Parallel I/O Buffer Attributes (TR)

The supplied parallel I/O library may have user configurable buffer lengths and all buffers associated with a job may be automatically flushed upon completion or failure of the job. It may also have an API for manual flushing of individual buffers.

5.2.8.3.2 Parallel I/O Sequential Write (TR)

Output from a cluster wide parallel job directed to stdout/stderr may be atomic on a line by line basis and be prefixed with a label indicating which thread or process performed the write.

5.2.8.3.3 High-Level Parallel I/O API (TR)

The supplied parallel I/O library may include an interface to a self-describing data-format, such as netCDF, or HDF.

5.2.8.4 Graphical User Interface API (MR)

The Subcontractor shall provide the standard X11R5 and Motif 1.2, or current versions, applications, servers and API libraries.

5.2.8.5 Math Libraries (TR)

The Subcontractor may supply highly optimized mathematical libraries, callable from all baseline languages, for serial (single processor) SMP and cluster wide applications.

5.2.9 SST Software Environment Features for ID (TR)

The ID system may present a similar general software environment as the SST does. The University is aware that the specifics (e.g., clustering software, intra-SMP code development tools, software scalability) between the two systems may differ. Indicate which of the SST Software target requirements in section 4.2 may be delivered with the ID (or shortly thereafter as part of a software technology refresh program).

5.2.10 Compliance with DOE Security Mandates (MR)

DOE Security Orders have changed over time. From time to time these mandates cause LANL and LLNL to fix bugs or implement security features in vendor operating systems and utilities. We require, for computer security purposes only on this contract, that the Subcontractor either provide to the University the operating system and utilities source code on a demand basis or make needed enhancements or bug fixes as required.

5.2.11 Cluster Environment On-Line Documentation (TR)

The Subcontractor may supply hardcopy and on-line documentation for all major hardware and software subsystems. The on-line documentation may be readable by all users utilizing an X11-based graphical user interface and standard terminal interfaces.

5.2.12 ID Applications Development Support (MR)

The Subcontractor shall supply at least one on-site analyst to provide expertise to the University code development teams in the areas of software development tools, parallel applications libraries and applications performance.

5.3 Performance of the ID System

The seven benchmark programs described below may be executed by the Subcontractor for the purpose of measuring the execution performance and compiler capabilities of the ID system. Each of the benchmark programs represents a particular subset and/or characteristic of the expected ASCI workload, which consists of solving complex scientific problems using a variety of state-of-the-art computational techniques. The general requirements and constraints outlined below shall apply to all of the benchmark codes. Additional requirements and/or constraints found in individual benchmark readme files shall apply to that individual benchmark.

5.3.1 Benchmark Suite (TR)

The test programs are available via the Web at http://www.llnl.gov/asci_benchmarks. There is a master readme file which provides further information regarding the benchmark suite as a whole. The individual benchmark codes can be downloaded as tar files. A readme file for each benchmark provides more detailed information about that benchmark including a more complete description of that benchmark, how to build and run it, and any specific constraints that apply to it. Access to some of the benchmarks is restricted and will require contacting the Contract Administrator to gain access.

The sPPM benchmark is an explicit three-dimensional hydrodynamics code designed to model problems involving high-speed hydrodynamic flow using the piece-wise parabolic method.

The SWEEP3D benchmark is a three-dimensional solver for the time-independent, neutral particle transport equation on a Cartesian mesh. The first-order form of the transport equation is solved by sweeping through the spatial mesh along discrete directions (ordinates).

The COMOPS benchmark is a communications program to test intra-cluster communications capability. Node-to-node, global reduction and mesh-oriented communication performance are measured.

The SMPT benchmark is a set of programs that tests the computational performance of a single SMP using various computational constructs. The set consists of three programs in three different languages (Fortran 77, Fortran 90 and C). Measurements of the time and MFLOP/s rate for each computational construct are recorded.

The DSBENCH benchmark solves Maxwell's curl equations in the time-domain and in three spatial dimensions. The code tests the use of unstructured non-orthogonal grids. Techniques in DSBENCH can be used with arbitrary unstructured grids composed of convex polyhedral cells.

The PCTH benchmark is an Eulerian hydrocode used to model time-dependent large-deformation material motion due to strong pressure impulses and/or large impact velocities. It is a large multi-language code written in C++, C, and Fortran 77.

The LFK benchmark is a suite of computational kernels that measures the computational performance of a single processor. The geometric mean of the rates of these kernels is reported.

5.3.2 System Configuration (MR)

If the Offeror chooses to submit benchmark results then those runs shall be made on a system with the following characteristics. The reference benchmark system shall be a scaled down version of the actual proposed ID system. The sPPM, SWEEP3D, and PCTH benchmarks shall be run on a 30 peak GFLOP/s configuration consisting of at least two SMPs. The DSBENCH program requires only a 10 peak GFLOP/s system but again consisting of at least two SMPs. The COMOPS program shall be run on an 8 SMP cluster although each SMP need not be fully populated with processors; the specific requirement and alternatives are discussed in the COMOPS readme. The SMPT benchmark requires only a single SMP because it is scaled relative to the full proposed ID system. The LFK program runs on only a single processor.

Reference Benchmark Configurations	
sPPM	30 Peak GFLOP/s 2+ Node Cluster

SWEEP3D	30 Peak GFLOP/s 2+ Node Cluster
PCTH	30 Peak GFLOP/s 2+ Node Cluster
DSBENCH	10 Peak GFLOP/s 2+ Node Cluster
COMOPS	8-Node Cluster
SMPT	Single SMP
LFK	Single Processor

The benchmark system should contain the same processors, cache, memory, SMP size, interconnects, etc. that the ID system would have. If this condition is not possible, benchmark results from an alternative system may be reported but the Subcontractor shall also provide estimated scaled performance to the requested configurations consistent with the benchmark system configurations as identified in the previous paragraph. All scaling arguments shall be fully described by the Subcontractor and will be reviewed and evaluated by the University; supporting documentation may be provided. The University will be the sole judge of the validity of any scaled results.

5.3.3 Test Procedures (MR)

If the Offeror chooses to submit benchmark results then those runs shall be made according to the following test procedures. The ASCI systems will be primarily used in a high-level language environment. It is the intent of these benchmarks to measure performance of the ID system from this standpoint. Recoding of the benchmarks or portions of the benchmarks in assembly language is prohibited. The use of library routines that currently exist in a Offeror’s supported set of general or scientific libraries, or will be in such a set when the ID system is delivered, is allowed at the University’s discretion when they do not specialize or limit the applicability of the benchmark nor violate the measurement goals of the particular benchmark. Source preprocessors, execution profile feedback optimizers, etc. are allowed as long as they are, or will be, available and supported as part of the compilation system for the ID system. All benchmarks that use the message-passing programming paradigm shall use a supported communication library that implements the MPI standard. Some specific constraints about the use of library routines apply to individual benchmarks as stated in individual benchmark readme files.

Changes to accommodate unique hardware and software characteristics of a system that are consistent with the preceding paragraph will be allowed except where specifically prohibited in the constraints for each benchmark. Code modifications shall be documented in the form of initial and final source files, preferably with accompanying text describing the changes. An audit trail shall be supplied to the University for any changes made to the benchmark codes. The audit trail shall be sufficient for the University to determine that changes made violate neither the spirit of the benchmark nor the specific restrictions on the various benchmark codes.

Output and measurements that shall be provided by the Offeror for each benchmark are:
1. the output of the results generated by each individual benchmark run,

2. any additional performance measurements as requested in the readme of each benchmark,
3. the CPU time, system time, and wall clock time for the entire execution of each individual benchmark run,
4. the maximum memory usage during the execution of each benchmark run, broken down by operating system usage and application usage,
5. all compilation options, the wall clock time required to compile all source code, and the wall clock time to load / link (i.e., create an executable image) for each benchmark.

Correct execution and measurements shall be certifiable by the University.

End of Section 5

6.0 Implementing a Sustained Stewardship TeraFLOP

Because of the complexity of this activity, a very strong project plan is of great importance. The Subcontractor's understanding of our requirements, approach to meeting those requirements, commitment of resources, and attention to cost are critical to the success of the project. In the same vein, the approach to managing this activity is critical. The need to have the support of corporate senior management and a major commitment to a quality assurance plan are also examples of areas critical to the success of the project.

The specific mandatory detailed planning and effort tracking and documentation requirements for the development and manufacturing efforts that shall be delivered as part of the contract are delineated in sections 6.1 and marked with (MR) designation.

The specific mandatory delivery milestones for the project that shall be met are delineated in section 6.2 with (MR) designation.

In addition to the mandatory delivery milestones, the Subcontractor may deliver any target delivery milestones, as delineated in section 6.2 with (TR) designation, and any Subcontractor-proposed features consistent with the objectives of this project and Subcontractor’s Research and Development Plan which the Subcontractor believes will be of benefit to the project.

6.1 Detailed Project Plan

This project envisions a quantum advance in delivered performance capability for LANL and LLNL scientists and engineers. To successfully reach this level of delivered performance the Subcontractor shall submit, within thirty (30) days of contract award, a detailed, highly focused plan delineating the research and development activities to achieve the project goals. At a minimum, the detailed plan shall contain the following components: management; hardware; software.

6.1.1 Detailed Project Management Plan (MR)

The Subcontractor shall submit for University approval, within thirty (30) days of contract award, a detailed project management plan. The plan shall contain at least the following components:

- Actual management team and structure. List the actual members of the management team, provide their resumes and indicate their roles and responsibilities. Provide an actual organizational chart of the management team and lines of reporting to various parts of the company. Indicate the company interface mechanisms with the University.

- Actual organization for core team. List the contributing organizations within the company and how they will be coordinated. Provide an organizational chart of the company that depicts these groups and their lines of responsibility. Include hardware R&D, software R&D, productization, field team and applications support, manufacturing, purchasing and quality assurance. Indicate how these areas will be coordinated by the management team.

- Detailed project plan and schedule. Provide a Work Breakdown Structure (including milestones) for the project giving at least five levels of detail, as appropriate, with projected start and finish dates and interdependencies of deliverables. This project plan shall elaborate on the tasks and milestones committed to in the scalable systems development section and clearly delineate the project critical path tasks (see below). Provide a Project Schedule that starts at contract award and ends with SST TeraFLOP/s sPPM Demonstration. The schedule shall be developed using the Critical Path Method (CPM) scheduling technique and shall utilize the same numbering scheme as the Work Breakdown Structure. The Project Schedule shall be placed under configuration control to ensure that all project schedule updates are accomplished in a manner that preserves an audit trail from the original Project Schedule to the current schedule status. The Schedule shall contain sufficient detail to ensure that the University and the Subcontractor can measure progress on an appropriate number of milestones and tasks on any path or on parallel paths to measure progress and to determine the true critical path to project completion.

- Risk reduction plan. In order to meet the project goals and objectives in a timely manner, indicate fall-back strategies that will become operative should accelerated delivery schedules not proceed as rapidly as predicted. Indicate additional resources that will be available to the effort in the event that problems develop. Delineate the problem escalation and resolution path. Designate and identify an individual within your company, whose line of reporting is outside that of the Project Management Team and Core Team, to provide independent quality assurance for this project. This individual shall report directly to the corporate executive ultimately responsible for this project and shall provide independent analysis of project schedule and potential problems.

6.1.2 Detailed Hardware Project Plan (MR)

The Hardware Project Plan shall contain at least the following components:

- CPU Technology. Identify the planned milestones for processor development that lead to those to be deployed in the SST. In particular, provide milestones for silicon process development, sampling, engineering quantities, production quantities for each processor generation(s) between the ID and SST.

- Shared Memory Subsystems. Provide the planned tasks and milestones for shared memory subsystem research and development. Include tasks and milestones for at least the following development areas: memory architecture; cache coherency protocols; interconnect technology; performance modeling efforts; applications analysis.

- SMP Product Development. Provide the planned tasks and milestones for SMP product development between the ID and SST system generations. Include milestones for productization, beta-test and general availability areas. Indicate how and when this technology would be inserted at LANL or LLNL in the technology refresh program.

- Cluster Interconnect. Provide the planned tasks and milestones for cluster interconnect research and development between the ID and SST system generations. Include tasks and milestones for at least the following development areas: increasing bandwidth; low latency message passing; interconnect topology; virtual memory DMA between SMPs; extending cache coherent distributed shared memory between SMPs. Indicate how and when this technology would be inserted at LANL or LLNL in the technology refresh program.

- Cluster I/O and External Network Connections. Provide the planned tasks and milestones for development of cluster I/O and external networking. By external networking, we mean the standards-based networking (e.g., HIPPI, ATM, Fibre Channel) to connect the cluster to other networks at LANL and LLNL. Include tasks and milestones for at least the following development areas: parallel I/O development; remote file systems in support of the RED/BLACK I/O partitioning scheme; external networking. Indicate how and when this technology would be inserted at LANL or LLNL in the technology refresh program.

- Scalable Systems Capability. Provide the planned tasks and milestones for the development and testing of scalable system components. Include development of hardware for reliability, availability and serviceability (RAS).

- Other. Provide the planned tasks and milestones for research and development efforts related to this contract that are not covered by the preceding requirements.

6.1.3 Detailed Software Project Plan (MR)

The Software Project Plan shall contain at least the following components:

- Operating System Development. Provide the planned tasks and milestones for SMP operating system development. Include tasks and milestones for at least the following development areas: scalable SMP support; distributed shared memory locality of reference; support for hardware and system performance monitoring; low latency user callable thread mechanism; memory management; full 64-bit support, journaled file systems; reboot time minimization; SMP local gang scheduling; virtual memory support for batch processing (e.g., process swapping and checkpointing); support for RED/BLACK I/O partitions; DCE standard tracking; group routing.

- Single System Image Development. Provide the planned tasks and milestones for cluster wide single system image development. Include tasks and milestones for at least the following development areas: POSIX compliant cluster wide file system; cluster single IP network address; page leveled shared memory between SMPs; file system failover; cluster wide lock management; cluster wide process and POSIX session ID space; virtual memory DMA between SMPs; load balancing and process migration between SMPs; system administration tools for managing the cluster as a single system;

- Parallel I/O Development. Provide the planned tasks and milestones for cluster wide parallel application I/O development. Include tasks and milestones for at least the following development areas: networked connected parallel I/O development; application programming interface; performance.

- Compiler Development. Provide the planned tasks and milestones for base language (Fortran 90, C, C++) and HPF development. Include tasks and milestones for at least the following development areas: mixed language support; automatic and directed parallelization (decomposition) of applications; latency reduction techniques; compiler optimization; migration support (from ID to SST).

- Message Passing Environment. Provide the planned tasks and milestones for message passing development. Include tasks and milestones for at least the following development areas: bandwidth and latency targets for MPI; MPI standard tracking; integration with debuggers, profilers and performance analysis tools; interoperability to cluster external resources.

- Code Development Tools. Provide the planned tasks and milestones for code development tools development. Include tasks and milestones for at least the following development areas: parallel make; profilers, debuggers, application performance monitoring tools, GUI development for code development tools.

- Cluster Resource Management Support. Provide the planned tasks and milestones for resource management development. Include tasks and milestones for at least the following development areas: hooks for external policy modules; system monitoring tools; cluster wide gang scheduling.

- Fault Tolerance and Containment. Provide the planned tasks and milestones for the development of fault tolerance and gradual degradation of service in the face of component failure features. Include tasks and milestones for at least the following development areas: failure tolerance of CPUs, memory components; SMPs, inter-connect, I/O subsystems; error detection vs. retry.

- Other. Provide the planned tasks and milestones for research and development efforts related to this contract that are not covered by the preceding requirements.

6.2 Project Milestones

Because of the need to meet Science-Based Stockpile Stewardship goals as quickly as possible, the project schedule and milestones are of critical importance. Meeting the following milestones is critical to the success of the project; earlier is much better.

The implementation will entail the installation of local clusters at Laboratory sites. Each cluster will be assembled from individual SMPs that are interconnected with a high-speed, low-latency interconnect supplied by the Subcontractor. These clusters will be connected to the site's local campus network and to the wide area network that interconnects the three Laboratories. Access to the resources will be provided locally via the site's existing campus networks and remotely through the ASCI-supplied WAN.

6.2.1 Detailed Project Plan (MR)
The Subcontractor shall provide a detailed Project Plan 30 days after contract award.

6.2.2 Initial Delivery (ID) System (MR)
The Subcontractor shall install and support an initial code development system containing 50 Gigabytes (GB) of memory, 2.5 TB of disk configured for RED/BLACK operations and capable of 100 GigaFLOP/s (GFLOP/s) of peak performance at LANL or LLNL, as directed, 90 days after contract.

6.2.3 Second Initial Delivery (ID) System (MO)
At the option of the University, the Subcontractor shall install and support a second initial code development system containing 50 Gigabytes (GB) of memory, 2.5 TB of disk configured for RED/BLACK operations and capable of 100 GigaFLOP/s (GFLOP/s) of peak performance at LANL or LLNL, as directed, 90 days after contract award.

6.2.4 ID Applications Development Support (MR)
The Subcontractor shall supply at least one on-site analyst to provide expertise to the University code development teams in the areas of software development tools, parallel applications libraries and applications performance at the time of ID system delivery.

6.2.5 FY97 Plan and Review (MR)
The Subcontractor shall provide a detailed plan of activities and deliverables for fiscal year 1997 for University review and approval in the first quarter of FY97.

6.2.6 Technology Refresh (TR)
The Subcontractor may provide hardware and software technology updates between the installation of the ID system and the installation of the SST that would significantly improve the hardware and software environment. Hardware technology updates may meet or exceed the following component scaling parameters:

- Memory Size/Peak FP (Byte/FLOP/s) ≥ 0.5
- Disk Space/Peak FP (Byte/FLOP/s) ≥ 25
- Cache Bandwidth/Peak FP (Byte/s/FLOP/s) ≥ 4
- Memory Bandwidth/Peak FP (Byte/s/FLOP/s) ≥ 1

- Intra-Cluster Network Bi-Section Bandwidth/Peak FP (Bits/s/FLOP/s) ≥ 0.5
- System Peak Disk I/O Bandwidth/Peak FP (Byte/s/FLOP/s) ≥ 0.03

6.2.7 FY98 Plan and Review (MR)

The Subcontractor shall provide a detailed plan of activities and deliverables for fiscal year 1998 for University review and approval in the first quarter of FY98.

6.2.8 SST Applications Development Support (MR)

The Subcontractor shall supply at least two on-site analysts to provide expertise to the University code development teams in the areas of software development tools, parallel applications libraries and applications performance at the three months prior to the SST system delivery.

6.2.9 Scalable Development Environment Demonstration (TR)

The Subcontractor may demonstrate the scalability of essential software capabilities in the application development environment across the clustered SMP system in mid CY 1998. Specifically, the Subcontractor may use the sPPM demonstration code on the full cluster to demonstrate debugger, event tracing and performance statistics capabilities.

6.2.10 Sustained Stewardship TeraFLOP (SST) Demonstration (TR)

The Subcontractor may demonstrate the SST scalable cluster in mid CY 1998 containing 0.5 Terabytes (TB) of memory and one (1.0) TeraFLOP/s of sustained performance on the sPPM demonstration code. The sum of peak and sustained performance of the SST scalable cluster shall be at least four (4.0) TeraFLOP/s.

6.2.11 Scalable Development Environment Demonstration (MR)

The Subcontractor shall demonstrate the scalability of essential software capabilities in the application development environment across the clustered SMP system no later than the end CY 1998. Specifically, the Subcontractor shall use the sPPM demonstration code on the full cluster to demonstrate debugger, event tracing and performance statistics capabilities.

6.2.12 Sustained Stewardship TeraFLOP (SST) Demonstration (MR)

The Subcontractor shall demonstrate the SST scalable cluster no later than the end CY 1998 containing 0.5 Terabytes (TB) of memory and one (1.0) TeraFLOP/s of sustained performance on the sPPM demonstration code. The sum of peak and sustained performance of the SST scalable cluster shall be at least four (4.0) TeraFLOP/s.

6.2.13 SST Installation (MR)

The Subcontractor shall install and support this system at either LANL or LLNL, as directed, for at least two years following a successful demonstration.

6.2.14 FY99 Plan and Review (MR)

The Subcontractor shall provide a detailed plan of activities and deliverables for fiscal year 1999 for University review and approval in the first quarter of FY99.

6.2.15 Memory Installation (MO)

The Subcontractor shall, at the University's option, install SST system memory up to an additional 1.0 TB by the first quarter of CY 1999. If the University chooses to exercise this option, installation of the additional memory before or after SST installation shall be at the Subcontractor's discretion.

6.3 Performance Reviews (MR)

Quarterly performance reviews shall be conducted between the Subcontractor's corporate executives and the University. The Subcontractor shall submit a Quarterly Project Status Report at least five working days before each quarterly review. The report shall provide the status of all work breakdown structure tasks and milestones in the critical path. It shall also contain narrative descriptions of anticipated and actual problems, solutions, and the impact on the project schedule. Numbered action items shall be taken, assigned, logged, and tracked by the Subcontractor. The minutes of all project reviews shall be recorded in detail by the Subcontractor and provided to the University for approval within 5 working days after the review.

6.4 SST TeraFLOP/s sPPM Demonstration (MR)

The sPPM demonstration code is of special interest to ASCI because it solves important hydrodynamics problems using an aggressive and demonstrated highly-efficient SMP cluster implementation of numerical methods relevant to DOE's Stockpile Stewardship Program. The sPPM demonstration code represents the current state of an ongoing effort which has demonstrated good processor performance, excellent multitasking efficiency, and excellent message passing parallel speedups all at the same time. Its use as an SST demonstration is to validate the ASCI effort by demonstrating a sustained one (1.0) TeraFLOP/s computation across all processors of a cluster of SMPs for a single large application which is of direct interest to the DOE Stockpile Stewardship Program as well as to scientific simulation in general. It is expected to bring recognition to the ASCI Program, to the Subcontractor and the University, as well as to the larger scientific high performance computing community.

The Subcontractor shall demonstrate a sustained one (1.0) TeraFLOP/s execution rate of the sPPM benchmark code across the entire SST system as required by 6.2.10 or 6.2.12. The University shall witness, verify and certify the demonstration. A sustained performance rate shall be computed only for the time step update portion of the sPPM code for a problem size of approximately 2000-cubed grid points for a number of time steps sufficient to run for at least one (1.0) hour of wall clock time. The exact problem size will necessarily be determined by the actual SST cluster size, the available application memory, and the achieved sustained computation rate. The computation rate shall be determined based on the actual executed operations as shown below and the elapsed wall clock time required to complete the time steps on the SST system.

+, -, *	1 FLOP each

/, sqrt	4 FLOPs
min, max, abs, sign	1 FLOP each

The sPPM one (1.0) TeraFLOP/s SST demonstration code shall be derived from the sPPM code from the ID benchmark suite. It shall be programmed in Fortran with some C and shall contain no custom assembly language. It may use either single or double precision IEEE arithmetic (or even a mixture). It shall use POSIX threads and MPI message passing. It may use general or scientific library routines if they are, or will be, part of a supported library. It shall use the initial conditions built into the ID code without the image or texture maps. The time for initialization, visualization and restart dumps will not be included in the one TeraFLOP/s rate determination - visualization and restart dumps could even be disabled. The timing prints per double time step can also be disabled but the one line “courant and energy” print is required on at least node 0 to either stdout or to the output file.

Optimizations will be allowed so long as they don’t specialize the functionality represented by the reference sPPM benchmark implementation. In particular, the source code for the hydro kernel of the sPPM benchmark code (i.e. the file sppm.m4) shall be used as provided by the University (i.e. unmodified), except for the addition of compiler directives. The University will continue its efforts to improve the efficiency of the code. Even for the subroutines in sppm.m4, tuning can be achieved through selecting alternate pieces of provided code through preprocessor flags, through the parameter IQ, etc. The goal is to emphasize higher level optimizations as well as compiler optimization technology improvements while maintaining “readable” code and physics modules. One obvious permitted modification in the parallel part of the sPPM demonstration code may be the overlapping of computation and communication (i.e., asynchronous communication).

Ideas to demonstrate Tera-scale performance levels on additional ASCI relevant applications may be suggested, but only if such opportunities arise and prove feasible and do not interfere with the sPPM SST demonstration should such endeavors be considered.

End of Section 6

Appendix A Glossary

Hardware

b	bit. A single, indivisible binary unit of electronic information.
B	Byte. A collection of eight (8) bits.
Cluster	A set of SMP's connected via a scalable network technology. The network shall support high bandwidth, low latency message passing. It may also support remote memory referencing.
CPU	Central Processing Unit or processor. A VLSI chip constituting the computational core (integer, floating point, and branch units), registers and memory interface (virtual memory translation, TLB and bus controller).
FLOP	Floating Point OPeration.
FLOPS	Plural of FLOP.
FLOP/s	Floating Point OPeration per second.
GB	GigaByte. GigaByte is a billion bytes. When used in terms of Random Access Memory, billion is 2^{30} (or 1,073,741,824) bytes. When used in any other context is 10^9 (or 1,000,000,000) bytes.
GFLOP/s	GigaFLOP / s. Billion (10^9 = 1,000,000,000) 64-bit floating point operations per second.
MB	MegaByte. MegaByte is a million bytes. When used in terms of Random Access Memory, million is 2^{20} (or 1,048,576) bytes. When used in any other context is 10^6 (or 1,000,000) bytes.
MFLOP/s	MegaFLOP / s. Million (10^6 = 1,000,000) 64-bit floating point operations per second.
NUMA	Non-Uniform Memory Access architecture. The distance in processor clocks between processor registers depends on where in main memory the address points to. That is, a load / store operation latency for some memory locations is larger than that for others.

Peak Rate	The maximum number of 64-bit floating point instructions (add, subtract, multiply or divide) per second that could conceivably be retired by the system. For RISC CPUs the peak rate is typically calculated as the maximum number of floating point instructions retired per clock times the clock rate.
Scalable	A system attribute that increases in performance or size as some function of the peak rating of the system. The scaling regime of interest is at least within the range of 100 GFLOP/s to 3.0 (and possibly to 10.0) TFLOP/s peak rate.
SMP	Shared memory Multi-Processor. A set of CPUs sharing random access memory within the same memory address space. The CPUs are connected via a high speed, low latency mechanism to the set of hierarchical memory components. The memory hierarchy consists of at least processor registers, cache and memory. The cache may also be hierarchical. If there are multiple caches, they shall be kept coherent automatically by the hardware. The main memory may be a Non-Uniform Memory Access (NUMA) architecture. The access mechanism to every memory element shall be the same from every processor. More specifically, all memory operations are done with load/store instructions issued by the CPU to move data to/from registers from/to the memory.
Tera-Scale	The environment required to fully support production-level, realized TFLOP/s performance. This environment includes a robust and balanced processor, memory, mass storage, I/O, and communications subsystems; robust code development environment, tools and operating systems; and an integrated cluster wide systems management and full system reliability and availability.
TB	TeraByte. TeraByte is a trillion bytes. When used in terms of Random Access Memory, trillion is (2^{40} = 1,099,511,627,776) bytes. When used in any other context is 10^{12} (or 1,000,000,000,000) bytes.
TFLOP/s	TeraFLOP/s. Trillion (10^{12} = 1,000,000,000,000) 64-bit floating point operations per second.
UMA	Uniform Memory Access architecture. The distance in processor clocks between processor registers and every element of main memory is the same. That is, a load/store operation has the same latency, no matter where the target location is in main memory.

Software

API (Application Programming Interface)	Syntax and semantics for invoking services from within an executing application. All APIs shall be available to both Fortran and C programs, although implementation issues (such as whether the Fortran routines are simply wrappers for calling C routines) are up to the supplier.
Current standard	Term applied when an API is not "frozen" on a particular version of a standard, but may be upgraded automatically by Subcontractors as new specifications are released (e.g., "MPI version 1.1" refers to the standard in effect at the time of writing this document, while "current version of MPI" refers to further versions that take effect during the lifetime of this contract.
Fully supported (as applied to sys-tem software and tools)	A product-quality implementation, documented and maintained by the HPC machine supplier or an affiliated software supplier.
Gang Scheduling	When a user job is scheduled to run, the gang scheduler must contemporaneously allocate to CPUs all the threads and processes within that job (either within an SMP or within the cluster of SMPs). This scheduling capability must control all threads and processes within the SMP cluster environment.
Job	A job is a cluster wide abstraction similar to a POSIX session, with certain characteristics and attributes. Commands shall be available to manipulate a job as a single entity (including kill, modify, query characteristics, and query state). The characteristics and attributes required for each session type are as follows: 1) interactive session: an interactive session would include all cluster wide processes executed as a child (whether direct or indirect through other processes) of a login shell and shall include the login shell process as well. Normally, the login shell process shall exist in a process chain as follows: init, inetd, [telnetd | rlogind | xterm | cron], then shell. 2) batch session: a batch session shall include all cluster wide processes executed as a child (whether direct or indirect through other processes) of a shell process executed as a child process of a batch system shepherd process, and shall include the batch system shepherd process as well. 3) ftp session: an ftp session would include an ftpd and all its child processes. 4) kernel session: all processes with a pid of 0. 5) idle session: this session does not necessarily actually consist of identifiable processes. It is a pseudo-session used to report the lack of use of resources. 6) system session: all processes owned by root that are not a part of any other session.

Published (as applied to APIs):	Where an API is not required to be consistent across platforms, the capability lists it as "published," referring to the fact that it shall be documented and supported, although it may be Subcontractor- or even platform-specific.
Single-point control (as applied to tool interfaces)	Refers to the ability to control or acquire information on all processes/PEs using a single command or operation.
Standard (as applied to APIs)	Where an API is required to be consistent across platforms, the reference standard is named as part of the capability. The implementation shall include all routines defined by that standard (even if some simply result in no-ops on a given platform).
XXX-compatible (as applied to system software and tool definitions)	Requires that a capability be compatible, at the interface level, with the referenced standard, although the lower-level implementation details may differ substantially (e.g., "DFS-compatible" means that the distributed file system shall be capable of handling standard DFS requests, but need not conform to DFS implementation specifics).